CANMET

Canada Centre
for Mineral
and Energy
Technology

Centre canadien
de la technologie
des minéraux
et de l'énergie

Energy Research Laboratories

Laboratoires de recherche sur l'énergie

Canada

Energy, Mines and
Resources Canada

Énergie, Mines et
Ressources Canada

Development of Process Control Software

Final Report by

Atlantis Scientific Systems Group Inc.

Work on the project was conducted under the auspices of
the Canadian Centre for Mineral and Energy Technology,
Energy, Mines and Resources Canada.
This work was supported by the Federal Panel on Energy R&D (PERD).

DSS FILE NO.: 003SQ.23440-8-9044
DSS CONTRACT SERIAL NO.: 23440-8-9044/01-SQ
DSS FILE NO.: CANMET PROJECT NO.: 501202
CANMET SCIENTIFIC AUTHORITY: T. J. W. de Bruijn
DATE: November 6, 1989

13 pp

MICROMEDIA

90-02546/

Executive Summary of the Development of Process Control Software Contract No. 23440 8 9044

Atlantis Scientific Systems Group Inc.
1827 Woodward Drive, Ottawa, Ontario, K2C 0P9

October, 1989

This contract involved the further development of a process monitoring and control software package, developed by Atlantis Scientific. In addition, a higher performance computer was evaluated to replace the inadequate existing IBM XT computer. Also, an intelligent communications board, with current loop mode was installed in place of the previous hardware, thereby enhancing system reliability. The software changes were made in several stages, with each stage concluded by an on-site test. Both the background and foreground programs of PC Control were translated to use more recent language compilers, after which, new features were added to the software. These additions included changes above and beyond the specifications of the contract. With the use of virtual tags we added a capability to do simple mathematical manipulations on data. The existing graphing feature was improved, and a new real-time graphing feature was added. The command language was extended, and syntax checking incorporated. The code was modularized for the easy addition of device drivers, and monitoring and control functions. A system summary generating function was added. Several independent diagnostics programs were developed to test the process control hardware and the device drivers. Finally, a new manual for PC Control was produced.

Sommaire du travail accompli pour le contrat numéro 23440 8 9044

Atlantis Scientific Systems Group Inc.
1827 Woodward Drive, Ottawa, Ontario, K2C 0P9

Octobre, 1989

Ce contrat porte sur l'expansion du logiciel de commande de processus, logiciel qui a été conçu par Atlantis Scientific. En plus de l'expansion du logiciel, un système de qualité supérieur à l'ordinateur utilisé présentement, l'IBM XT, a été évalué dans le but de remplacer cette ordinateur devenu insuffisant pour la tâche. Pour rendre le système plus fiable, la carte de communication de l'ordinateur a été remplacée par un circuit intelligent, doté d'une boucle de courant. Les changements apportés au logiciel ont été faits par étape. Et après chaque étape, ces changements ont été soumis à des essais pour s'assurer du bon fonctionement du programme. Tous les programmes destinés à la commande du processus de l'ordinateur ont été traduits pour permettre l'emploi de compilateur pour langage de programmation plus moderne. Une fois cette transformation accomplie, des améliorations ont été ajoutées aux programmes. Certaines de ces améliorations représentent de changements au delà des stipulations énoncées dans le contrat. La possibilité de faire des manipulations mathématiques simples avec les données a été ajoutée au logiciel grâce à des symboles illusoire. La capacité de l'ordinateur pour faire des graphiques a aussi été ameliorée et la possibilité de tracer des graphiques en temps réel a été ajoutée. Le langage de commande a été enrichi et la capacité de vérifier la syntaxe des instructions a été incorporer au système. Les instructions du logiciel ont été divisées en module pour faciliter l'addition d'instructions pour la commande périphérique et pour mieux permettre au logiciel de contrôler l'ordinateur. Une expansion du logiciel a aussi été conçu pour donner à l'ordinateur la capacité de produire un sommaire qui indique quel appareil est branché sur quel machine en tout temps. Des programmes permettant de diagnostiquer l'équipement de la commande de processus et de la commande peripherique ont été élaborés. Finalement, un manuel pour le logiciel de commande de processus a été écrit.

Final Report of the Development of Process Control Software
Contract No. 23440 8 9044

Atlantis Scientific Systems Group Inc.
1827 Woodward Drive, Ottawa, Ontario, K2C 0P9

October, 1989

This document is intended to be a description of work done on each phase, and provides information on where contractual obligations were exceeded to add to the usefulness of the software.

Phase 1: Specification for a new microcomputer

A written report of our findings was presented after researching the computer market at that time. The computer specification was dependent on efficiently running the following process control software modules: i) the user interface program, PCC32ABC and ii) the background monitoring and control program, CONTROL, and several other support programs. As a result of our suggestion, an 80386-based Compaq, model 386/20 was purchased.

Other phases of the contract dealt with re-writing of these software modules to be more robust, with the use of the latest programming tools and method, and the incorporation of useful enhancements. Also, some communications hardware was subject to change. It was important to bear these changes in mind when selecting the new computer. All process control software and all data manipulation programs were found to run on the new computer unaltered.

P. C. Control for the hydrocracker reactor made use of two special hardware plug-in boards: A multiple serial communications board, Digicom8, and two counter timer boards, Atlantic Scientific's model CTC100. Both these had to operate as is, or with a minimal amount of modifications in the higher performance microcomputers. When we finally installed these two boards into the new computer, all these boards worked as before, with only one simple modification to the pair of CTC100 boards.

Typically, a spread-sheet-type program, such as Lotus Symphony was to be used for simple data processing of the process-generated data. It had been observed that Lotus Symphony operates quite inefficiently with CONTROL, running simultaneously in the background. In so far as the processor throughput is concerned, a large speed-up (factor of about 10-20) has been achieved with the new computer.

The reliability of the new computer was one of the chief concerns. Although our specification was for a very reputable manufacturer of computers, Compaq, there was an unexpected malfunction with the main board of the computer and an intermittent fault in the video graphics board.

Both of these problems were attended to by the supplier within a reasonably short time. Atlantis Scientific also provided the CANMET group a loaner video graphics board while they were awaiting a replacement for the faulty board. Since the initial hiccups, the computer hardware has worked normally for extended periods of time.

The issue of future additions of computer peripherals was anticipated when specifying the computer. Indeed, some input/output slots are available for this purpose. In addition, being a 386-based computer, future versions of software, whether for control or data manipulation can make use of its 80386 code capability. While the current system has 1 megabytes of memory, expansion upto 13 megabytes is possible, without the use of an extra slot. The choice of an EGA- compatible board has allowed the use of colour graphics for PC Control software.

Following the purchase and delivery of the new computer, all control software was transferred over from the old computer. Acquisition and communications boards were also installed into the new computer. Then, PC Control was tested and found working normally.

Phase 2: Conversion of the User Interface Program

Phase 2 of the contract comprised the conversion of the previous two user interface programs into a single program. This was to be achieved with the then available Pascal language compiler, Turbo Pascal 4.0[1]. While older Pascal compilers only allowed smaller programs, the new version gave us the necessary tools to combine the two programs into one. This task was completed as intended, with as few alterations to the code as possible. The new program, PCC, version 2.9, had no enhancements added at this stage, as per the description in the contract. All user functions were tested, one by one, after completion. The previous background CONTROL program, version 2.0 was used for testing purposes. All source code for PCC version 2.9 was delivered as files on a floppy disk at the conclusion of the task.

Phase 3: Conversion of the Background Program

Phase 3 of the Process Control Software Development comprised of the translation of the background control program from the old DeSmet C language compiler to the industry standard Microsoft C language compiler. The objective was to do a direct translation, with no enhancements added at this stage. This phase began on May 2, 1988, and was allocated 30 working days. However, due to the various difficulties that arose in the translation and testing, and from our desire to build PC Control to the highest standards, it was completed in 48 days, 18 days in excess.

Testing of the new CONTROL program was facilitated by checking and correcting all subtle "programming language misuse", which would normally be ignored, and by the use of debugging tools from Microsoft, which helped eliminate other minor program faults. The new version was installed and field-tested feature by feature at all three CANMET sites.

---

[1] The final version of the software was coded with the improved Turbo Pascal 5.0 compiler

Phase 4: Training and Test Support

Under this phase, training and test support for PC Control was provided as and when required. The final version of the software was installed and tested under this phase.

Phase 5: Enhanced PC Control Features

Improvements and new features for PC Control were to be added under this phase. All new features were grouped under several tasks, as described below. Details on how the implementation was carried out are presented.

Task 1 and 2: On line calculations and calculated variables

It was intended to calculate feed rates, off gas rates, temperature differences, mass balance values on line, as well as have display and storage facilities for them. We implemented this phase in a more general way, so that mathematical calculations on tag values could be programmed by the user, as desired, in what are known as 'virtual' tags. These are a form of the normal level input type tags, thereby allowing all the normal monitoring, alarming, graphing and logging capabilities for these tags. Upon completion of this task, virtual tags were found to be of great use. With their use, and a new feature of PC Control, the filtered mode, a means was found for controlling the feed rate of stock into the system. This used to be one of the main obstacles in maintaining unattended operation. It is to be noted that both Task 1 and Task 2 were answered by the implementation of virtual tags for PC Control.

Task 3: Improved graphing facilities

The next task for enhancements was the improvement of the plotting facility of PC Control. This together with the two tasks above were grouped into phase 5A, which were written, and delivered as: 1) version 3A of PCC user interface, 2) version 1 of LMACROS macro loader, and 3) version 3A of CONTROL background control program.

The new trending facility was designed to allow upto six variables on one XY plot. Each trace can have a different programmable range, with y and time axis scales calibrated with appropriate values. Each trace is displayed in a different colour and line style. Use was made of as much vertical size as possible, so that even very small variations in data values could be easily resolved. Owing to the many different graphics display boards available today, a strategy was used that automatically detects the type of board and the pixel resolution available. In addition to this, a complementary graphing function was added that displays data values in real time (see Phase 10).

When all three tasks of Phase 5A were completed, the programs were thoroughly tested in-house, and then an additional period of 24 hours of on-site-testing was also conducted.

Task 4: Alarm Log (and Error Log)

This task entailed the addition of an alarm trail log, which would indicate to the operator when a point goes into alarm, and when it comes back to normal operating conditions. Each alarm event records the date, time, tag name, and its data value, and the type of alarm condition that occurred. We not only implemented an alarm log, but an 'error log' as well. The error log functions in a similar way to the alarm log, by recording every system error, whether from hardware or software into a log, with a date and time stamp. Both these trail logs are saved to disk files, which are in ASCII format, and are appended every time an alarm or error occurs. Also, the operator is allowed to view the last 40 events in each log on the screen by pressing <F7> or <F8> keys, and to print them using the <PrtScr> key of the computer. Alternatively, any error or alarm log file can be printed by issuing the following command at the DOS prompt: `COPY \PCC\ALARMS\FILENAME.ALM LPT1:`.

The trail logs have been found to be extremely useful tools, not only for the operators, but for us, the software developers as well. The logs have provided a means of tracking down problems with the entire system; When the cause is software related, it may be due to an error in typed action commands, or with the data acquisition boards, (hardware related). In each case, PC Control pinpoints the type of error, and the cause. For example if a syntax error occurs, both the inputted action command and the syntax error are shown. Related to the error log is the new PC Control's syntax checking facilities, which will be described below. As part of task 5, error messages issued as a result of hardware problems were greatly improved and their number increased. All possible error messages have been described in the appendix section of the PC Control manual. The alarm logs are also useful in determining plant failures that occur when no operators are present. In the past, an alarm event resulted in an alarm action command being executed, usually a corrective or shut down procedure, but the operator was not aware of when and why this occurred.

Task 5: Extended Command Language and Other Improvements

PC Control has a simple command language that allows the operator to achieve process changes, without having to use the menu system. This task involved adding new commands and improving the command entry into the system and error messages. Other improvements that were specified included a file loading and saving function for 'time queues', and data acquisition into trend buffers via the background program, instead of the foreground program.

Internally, the software has a 'parser' which deciphers the typed action commands. In order to add more commands and improve command entry and report syntax errors more effectively, the parser was re-written. The process of re-writing the parser was an investment in time, since it is a complicated part of the software. However the objective was to make the addition of new commands easy, which we achieved handily. At the same time, new features such as more explicative error messages, an ability to use any number of spaces, and the use of a comma ',' to separate commands were added. With the new parser:

- Any number of spaces can be used between words of a command
- Set points can be incremented by a positive or negative amount using the `SET+` command

- A time delay in macros can be achieved with the `WAIT` command
- Alarm limits for level input and state input tags can be changed via new action commands - see PC Control manual, page 80
- PID Control parameters, namely the P, I and D gains can be changed via new action commands - see PC Control manual, page 80
- Alarm monitoring for individual tags can be turned on and off with the `$tagname ALARMS ON/OFF` command
- Level input tags can be reset to read a given value using the `$tagname EQUALS newvalue` command

Beyond the requirements of the contract, the following useful action commands were also added:

- PC Control's reporting function can be toggled using the `REPORTING ON/OFF` command
- PC Control's logging function can be toggled using the `LOGGING ON/OFF` command
- the interval between log outputs can be set using the `FREQUENCY LOGGING minutes` command
- the interval between report outputs can be set using the `FREQUENCY REPORTING minutes` command
- the data acquisition rate into the trend buffers can be set using the `FREQUENCY TRENDING minutes` command
- the new syntax checking function can be activated using the `CHECK items` command

The ability to do syntax checking and automatic syntax checking are functions that were both addressed easily once the new parser was written. Now, every typed command is checked for correctness, and any errors are reported, both to the console, and to the error log. This applies to the tag editor, time queue editor and macro editor as well. It is now easy to check for any non-existing tag name or macro name for example, by doing a syntax check on the entire system. This is most useful before commencing a long, critical run. Many problems with previously written macros, which are a series of action commands, and with alarm action commands were brought to the attention of operators after this feature was is operation. Syntax checking brought to PC Control a level of intelligence, that further improves the system's reliability.

A time queue allows commands to be executed at preset date and times. A file editor for time queues was added, and at the same time, the menu structure improved so that each of the file editors was consistent in its use. The time queue editor allows one to not only create and edit a time queue, but to save it for later recall. When saved, the time queue has saved in it only relative time offsets, which when recalled become automatically adjusted to reflect absolute date and times in the future. Logic checking was added to the time queue editor so that all queue entries can only be in a chronological order.

With the use of virtual tags, as implemented in Tasks 1 and 2, new control schemes are possible. For example, control of a parameter can be dependent or the square of the difference of two temperatures, measured by different tags. Virtual tags allow simple mathematical operations on updating data obtained in real tags. However they are simply a level input type of tag and can therefore be used in control loops, making possible sophisticated control schemes. The control of feed rate has been achieved with the use of two virtual tags, one placed in a filtered mode, and another in the difference mode. The second virtual tag is a measure of the feed rate, which controls the feed pump setting. [2]

One very important enhancement to PC Control was transferring the data acquisition function from the foreground to the memory resident background program. This was achieved by adding a memory resident data storage area, and program code to update this trend buffer automatically. It is now possible to leave the foreground program, and return back to it, while the background continues data acquisition.

Phase 6: Modification of code structure

This phase involved the modification of the code structure to facilitate the addition of any other input/output subsystems, without affecting the integrity of the critical paths of the control program. Our desire to 'modularize' the software arose from a need to be able to add future data acquisition support and functions. This phase was implemented at various stages, as each section of the program was being modified for enhancements, or for translation into another language compiler. Most of the work involved modification of the background program, CONTROL, which is written in C language. One of the important tasks was to remove all hardware dependent code from the various modules into one specific module, `PCC_IO.C`. There are now three parts to the CONTROL program: the main monitoring and control code, the hardware dependent code in `PCC_IO.C`, and the 'device drivers', which are the actual software interfaces to the acquisition hardware units. Next the method of obtaining current time from the system was improved, and together with a change in the interrupt handler, a means was provided to code installation specific update intervals: 0.5 seconds for NRC's Engine Data Acquisition and Control System (EDACOS) was made possible this way. In fact, writing this entirely different version of PC Control was relatively easy once the version specific program code was 'conditionally compiled'. By conditionally compiled, we mean that just one set of program source code exists for all possible verions. Each of the different versions is created simply by stating some options at the time of compilation. The foreground, which is written in PASCAL language was also changed in places, so that different versions of PCC could be conditionally compiled for NRC or CANMET. In particular, all references to hardware devices are conditionally compiled.

The EDACOS version uses different acquisition and control hardware from what was previously used: a set of data acquisition boards made by Data Translation. Device drivers were written for boards DT2801 and DT2815. The final version for EDACOS has been installed and tested, and will be shown to the CANMET Scientific Authority to demonstrate the results of the code modification task.

---

[2] "Process Control of a Continuous Bench-Scale Hydrocracking Unit", T. J. W. de Bruijn, J. Patel and L. P. Carr, presented at the 39th Canadian Chemical Engineering Conference, October 1-4, 1989, Hamilton, Ontario, Canada

Phase 7: Inclusion of an Intelligent Multi-Port I/O Board

This phase required the addition of an intelligent (one having its own microprocessor) multi-port serial interface board. The objective was to reduce the current heavy I/O interrupt loading on the main processor, which was attributed to the cause of program malfunction. The existing system was arranged to use the two serial communications interrupts, one for COM1 and the other for COM2, which removed the option of remote communication capability.

All these issues were addressed by the replacement of the Digicom8 board and a serial communication board by a single intelligent serial communication board, Stargate model ACL-8. This board incorporates its own microprocessor for data handling, but uses none of the interrupts of the host computer. The model used in CANMET has 64K of memory. The Stargate board has been configured as shown below:

- The I/O address of the board is 200H to 203H, unless changed - see source file `PCC_STAR.C` for more information (switches sw2-1 OFF and sw2-8 OFF)
- The board operates with no interrupts to the computer (switchs sw2-2 to sw2-7 are all OFF)
- The computer in which this board operates is an AT type - jumper 2 is over pins 2 and 3
- The I/O bus of the computer runs at 8 MHz jumper 1 is over pins 1 and 2
- The dual port ram is mapped to D000 segment

There are 8 serial channels available on the Stargate board, and they have been assigned as follows:

Device	Channel	Baud	Parity	Data	Stop	Protocol	Rx Buffer	Tx Buffer
uMAC 4000	0	9600	ODD	7 bits	1 bit	NONE	4064 bytes	4064 bytes
Toledo WT51	1	4800	EVEN	7 bits	2 bits	NONE	73 bytes	4064 bytes
Toledo WT91	2	4800	EVEN	7 bits	2 bits	NONE	73 bytes	4064 bytes
Toledo WT151	3	4800	EVEN	7 bits	2 bits	NONE	73 bytes	4064 bytes
Toledo WT161	4	4800	EVEN	7 bits	2 bits	NONE	73 bytes	4064 bytes
Available	5	-	-	-	-	-	4064 bytes	4064 bytes
Available	6	-	-	-	-	-	4064 bytes	4064 bytes
Available	7	-	-	-	-	-	4064 bytes	4064 bytes

A general driver program for the Stargate board was written, called `PCC_STAR.C`. The program also contains testing modules which can be activated when the board or software need to be tested. Two drivers programs, `UMAC_SIO.C` and `TOLE_SIO.C` were written that call routines in the Stargate driver program. All three programs were thoroughly tested with separate test programs, that eventually became the diagnostics programs, noted in Phase 11.

Phase 8: Revision of the User Manual

Version 3.2 of PC Control software has a complete manual to describe it feaures. We used PCTEX, a publishing software program to produce high quality output. Ten copies of the final manual were delivered to the CANMET Scientific Authority.

Phase 9: Serial Communications: RS232 to Current Loop Conversion

This phase involved the conversion of all peripheral devices communicating presently in RS232 voltage levels to the 20mA current loop method. This change was intended to increase communications reliability; With the current loop method, it would allow distances of the order of 400 feet between the computer and peripheral communicating at 9600 baud. RS232 voltage level signals can operate at less than about 50 feet. This work was undertaken at the time of installing the intelligent serial input/output board, phase 7. All five devices, the uMAC and 4 Toledo weigh scales already supported the current loop mode of serial communications. Commercial RS-232 voltage to current converters from Altecon Data Communications were used at the computer end (for reference, see manual for "RS232 to Current Loop Converter". After experimenting with the converters, a deficiency was noted and corrected by wiring in extra electronics (power supply and resistors) into the original circuit of the converters. All five converters and extra circuitry was constructed and mounted in a box, which was later affixed to the wall.

The following is a table of wire connections at devices. Communications connections to the Toledo weigh scales is via the DB25 socket at the rear of the units. For each Toledo weigh scale, a resistor of 1.47 kohm is connected between the 22V power supply (pin 25) and the transmit wire, T+, to limit the current to about 20 mA. A jumper is connected between common output, pin 11 and signal ground, pin 12. Both the resistors and jumpers are in the DB25 shell connector to the weigh scale. The uMAC connections are to the P5 port at the left edge of the instrument.

Device	Wire	Connections at Device
Toledo WT51	Spare 1	Blu(Pin 10, T-) & Pur(Pin 25, T+)
Toledo WT91	Spare 1	Whi(Pin 10, T-) & Red(Pin 25, T+)
Toledo WT151	20	Whi(Pin 10, T-) & Red(Pin 25, T+)
Toledo WT161	10	Whi(Pin 10, T-) & Red(Pin 25, T+)
uMAC	'UMAC'	Whi(Pin B, PR+) & Red(Pin A, PR-)
uMAC	'UMAC'	Grn(Pin 2, KB+) & Blu(Pin 1, KB-)

The following is a table of wire connections in 20ma-to-RS232 converter box. The converter box houses 5 circuit boards, one each for the 5 devices. The circuit boards are Altecon Data Communications 20mA-to-RS232 converters, Model 232CL2[3]. Two 12 volt power supplies are also wired into the box. The pin numbers indicated below are connections from the current loop wires to the female end of the 232CL2 circuit boards. In the case of the uMAC, when the computer receives data, the internal 470 ohm resistor between pins 21 and 22 and the 12 volt supply are used to generate 20 mA current. This goes out to the uMAC, and returns via pin 25. Pin 23 of the LED is connected to the ground of the 12 volt supply. For the computer to transmit data, a 12 volt supply between pins 9(-12V) and 17(+12V) is connected. Pin 17 is the source of the 20 mA current loop, which returns via Pin 14. Pin 19 is connected to an internal resistor between pins 6 and 9. The female ends of the 232CL2 circuit boards which are the 20mA current inputs, are accessible from the top of the box. The male ends of the 232CL2 circuit boards provide the RS-232 voltage signals that are fed via short cables to the Stargate ACL8 outlet panel.

---

[3] See manual for Altecon Data Communications 20mA-to-RS232 converters, Model 232CL2

Device	Wire	Connections at Device
Toledo WT51	Spare 1	Blu(Pin 23, R-) & Pur(Pin 25, R+)
Toledo WT91	Spare 1	Whi(Pin 23, R-) & Red(Pin 25, R+)
Toledo WT151	20	Whi(Pin 23, R-) & Red(Pin 25, R+)
Toledo WT161	10	Whi(Pin 23, R-) & Red(Pin 25, R+)
uMAC	'UMAC'	Whi(Pin 22,) & Red(Pin 25, R+)
uMAC	'UMAC'	Grn(Pin 17, T-) & Blu(Pin 14, T+)

Phase 10: A real-time graphing facility

In addition to the trending facility that plotted 6 variables on one large XY plot, a real-time graphing facility was desirable. This would have 6 independent graphs on one screen. When implemented, the XY/bar graph facility was made into a much more sophisticated graphing facility than the trending, in that it updates the plots in real time. As well as the XY-mode of plots, the data can also be viewed as a bar graph, with set point, alarm limits and tag descriptors displayed along side the bar. Also shown on the bar graph format is control output information, being the level output tag name and its updating value.

The real-time graphing feature has greatly enhanced PC Control, for example, during the tuning of a new control loop. When estimating optimum tuning PID parameters, both the input and output signals can be viewed in real time. Quite often, it is imperative to have a direct indication of the plotting axis for variables of different types: temperature and weight measurements for example. This is achieved by the six individual plots. We were able to include this feature in an early version, 3A of the delivered software, for trial. Subsequently, this feature was refined and minor program faults corrected for the eventual design.

Phase 11: Stand alone diagnostics programs

Three programs were to be developed for testing each of the three hardware devices: the CTC100 boards, the Toledo weigh scales and the uMAC subsystem. These programs were written to make use of the same program modules that go into the background PC Control program. This allowed a complete test of not just the hardware, but the device driver software as well. Each of the programs is written with instructions to guide the user through the phases of the test. The programs are called: `UMACTEST.EXE`, `TOLETEST.EXE` and `CTCTEST.EXE`. They are kept in the D: drive, in directory `D:\MSC\DIAG`. To run any of these programs, the user simply types in the program name. It should be noted that the PC Control's background program must NOT be running at the same time. Through the use of the Microsoft C complier's debugging program, these programs can be single-stepped for more detailed testing[4].

Phase 12: An information reporting feature

To obtain system configuration information, the user required a way of producing summary reports on demand from within PC Control. Individual categories of information was also desired, with summary reports available for screen viewing or for printer output.

PC Control now incorporates the Info(rmation) function, which directs summary reports to printer or screen on one of the following categories:

1. Data Convert: Data conversion and update modes and rates
2. Set Points: Control loop information - input and output tags and set points
3. PID: PID gain constants, and control update rates
4. Alarms: Alarm limits, status and alarm action commands
5. Hardware: Hardware device, channel and bit numbers

We added the first item, 'data convert' to the original contract requirements. We have also added a date and time stamp to reports produced by the 'Info' function.

---

[4] Consult Atlantis Scientific if this type of testing is required.

Atlantis Scientific
Systems Group Inc.

Specialists in R&D Systems Development

CONTRACT SERIAL NO. 89044-01-SQ

CONTRACT REQUISITION NO. 23440-8-9044

CONTRACTOR: ATLANTIS SCIENTIFIC
SYSTEMS GROUP INC.

PC Control* User's Manual

MICROMEDIA

Atlantis Scientific Systems Group Inc.
1827 Woodward Drive, Ottawa, Ontario, K2C 0P9, Canada.
(613) 727 1087
Version 3.2, September 1989
*Developed in cooperation with CANMET/ERL and
licensed from Canadian Patents and Development Limited

102 pp

Contents

I Introduction and Installation 1

1 Features of PC Control 1

2 System Requirements 2

3 What's on Your PC Control Diskettes? 3

4 Installing PC Control 4

II Getting Started 5

1 Invoking the PC Control Program 5

2 Running the Demo Example 6

3 What the Screen Shows 6

4 What the Demo Process Represents 6

5 What the Symbols and Numbers Mean 8

6 What are Tags? 8

7 How the Menu Works 8

8 Purpose of esc key 9

9 How to Quit PC Control 9

10 How to Obtain Help 9

11 Exploring the Menus 11

12 Using the Command language 12

13 The Function Keys 13

14 Obtaining Graphs **14**

15 Screen or Schematic Editor **16**

16 How to Change Level and State Output Tag Values **16**

17 What is PID Control? **17**

18 Set Point Ramping **19**

19 Alarm System **19**

20 Macros, an Example **20**

21 Time Queues, an Example **21**

III Reference Section **24**

1 System Overview **24**
- 1.1 Introduction 24
- 1.2 Theory of Operation 26

2 Operating Modes for PC Control **29**
- 2.1 OFF Mode 29
- 2.2 MONITOR Mode 29
- 2.3 MANUAL Mode 30
- 2.4 AUTOMATIC Mode 30

3 Obtaining System `Info` rmation **31**
- 3.1 Setting Screen or Printer `Destination` 31
- 3.2 `Items` for System Information 31

4 Tags Editor **34**
- 4.1 Access to Tags Editor 34
- 4.2 Making a New Tag File 34
- 4.3 Editing tags 35
- 4.4 Special Level Input Tag Specifications 36
- 4.5 Special Level Output Tag Specifications 37

5 Tuning the System **39**
5.1 The Tune Menu 39
5.2 PID Control and Tuning the System 40
5.2.1 Setting up a Control Loop 40
5.2.2 Editing the PID Coefficients 41
5.2.3 Tuning the Loop 41
5.2.4 Changing Set Points and Ramping Set Points 43
5.2.5 Viewing Loop Response 43

6 Screen Editor **44**
6.1 Using the Screen Editor 44
6.2 Creating, Loading and Saving a Screen file 44
6.3 Screen Editing Functions 44

7 Action Commands and the Error Log **47**
7.1 Rules for Entering Action Commands 47
7.2 Syntax Checking, Error Messages and Error Log 47

8 Macros **50**
8.1 Using the Macros Editor 50
8.1.1 Creating, Loading and Saving a Macro file 51
8.1.2 Editing 51

9 Alarm System **53**
9.1 Introduction 53
9.2 Enabling Alarms 53
9.3 Viewing and Setting Alarm Trigger Points and Alarm Actions 54
9.4 Syntax Checking Alarm Action Commands 55
9.5 Alarm Limits 55
9.6 Alarm Log 56

10 Time Queue Operation **57**
10.1 Introduction 57
10.2 How Time Queue Operates 58
10.3 Time Queue Editor 58
10.4 Creating a Time Queue file 58

10.5 Using the Time Queue Editor 59
10.6 Chaining Time Queues 60
10.7 Repeating Time Queues 60

11 Data Graphing, Reporting and Logging **61**
11.1 **Setting Trending Interval** 62
11.2 **Graphing Facilities** 62
11.3 Setting up Graphing Parameters 62
  11.3.1 XY/Bar 63
  11.3.2 Historical Trending 63
11.4 **Reporting** 63
11.5 Setting up Reporting Parameters 65
  11.5.1 Enabling Reporting 65
  11.5.2 Setting Other Reporting Parameters 65
11.6 **Logging** 65
11.7 Setting up Logging Parameters 67
  11.7.1 Enabling Logging 67
  11.7.2 Setting the Logging Interval 67

12 Virtual Device Tags **68**
12.1 Introduction 68
12.2 Preparation 68
12.3 Entering the Mathematical Formula 69
12.4 Operands 69
12.5 Operators and Overflow Protection 69
12.6 Programming Examples 70

IV Appendices **73**

A Error Messages **74**
A.1 Hardware Related Error Messages 74
A.2 Software Related Error Messages 75

B Action Command Syntax **78**
B.1 System Commands 79

B.2 Tag Commands 80
B.3 Notes on Special Commands 81

C PID Control Algorithm **82**
C.1 Traditional Algorithm 82
C.2 Reset Windup Prevention 83

D Low Pass Filter Mode for LI Tags **84**

V The NRC-EDACOS Version **85**

E Installation Information **85**

F Panel/Remote Control and Bumpless Transition **86**

G Trend Log, Long Trend Log and Time Queue Log **88**
G.1 Special Action Commands 88
G.2 Trend Log 88
G.3 Long Trend Log 88
G.4 Time Queue Log 89

H Edacos Printout Facility **90**
H.1 Printout Format File 90
H.2 How to Obtain Data Printouts 90

I Transient Test Facility **91**
I.1 Creating Transient Files 91
I.2 Loading and Running Transient Test Sequence 92
I.3 How to Stop Transient Test Function 92

J Multi-Level Alarming **93**

Part I

Introduction and Installation

This chapter is included to assist first time users of PC Control in familiarizing themselves with its features and installing the software. The hardware requirements for PC Control are also outlined.

1 Features of PC Control

The PC Control system is a microcomputer based process controller designed to meet the needs of laboratory and pilot plant applications. PC Control can monitor and control as many as 100 process variables such as temperatures, pressures and flow rates. Monitored data can be displayed on the computer screen against a schematic representation of the process. PC Control can monitor input signals and control output signals (to heaters, motor drivers, solenoid valves, etc). Control of output signals is accomplished either 'manually' by entering commands into the user interface program, or 'automatically' via P.I.D. (proportional plus integral plus derivative) control action. In addition, output control can be achieved through 'time queue' actions and as a result of alarm tripping.

In addition to the formidable process control capabilities of this system, facilities for on screen plotting of signals, interval data reports and exporting data to Lotus© 123 or Symphony are all provided in real- time.

A summary of the capabilities offered by PC Control follows:

- A menu oriented user interface for ease of use
- Real-time data acquisition of up to 100 process inputs, with programmable modes for analog inputs (replace, accumulate, difference or low pass filter)
- Continuous display of process variables on process schematic
- Four programmable alarm levels for each analog (or level) input variable, and two for each digital (or state) input variable, with associated responses
- Manual or tunable PID control of analog (or level) output variables
- Automatic, real-time reporting of all process variables to printer or disk at programmable intervals
- Automatic, real-time logging of specific process variables to Lotus 123 number import format disk files at programmable intervals
- Acquisition of data at programmable rates for presentation as historical trend plots and as real time XY and bar plots
- A simple, but powerful command language to make process changes via actions, to by-pass using the menus

- Macros commands, allowing single word replacements of multiple command actions, with macro nesting capability
- Time queues containing scheduled command actions for execution at specific time and date, including the loading of (chaining to) another queue at the end of the current one
- Time queue and macro editors, allowing files to be created, loaded and saved
- Tag editor which specifies process variable's characteristics, and allows tag files to be created, loaded and saved
- Virtual, or calculated points with programmable math operations, and control and alarm capabilities
- System information, provides configuration summaries of your system by category, outputs to screen or printer
- Screen editor allowing process schematics to be designed or modified
- Alarm log of historical alarm events
- Error log of historical action command errors and I/O (input/output) device errors
- Automatic syntax checking on action commands, including complete system checks
- Two levels of on-line help including context sensitive help explaining most menu options, and an extended help facility providing general explanation of system features and operation.

2 System Requirements

For optimal operation of PC Control the following system requirements should be enforced. PC Control should be used on an IBM XT/AT/386 or compatible microcomputer with MS-DOS 2.0 or later, 640 KB of RAM, a CGA, EGA, VGA or Hercules graphics adapter and a hard disk with 20MB or more of storage. Ideally, an EGA or VGA board with a colour monitor should be used to make full use of the software's colour output capability.

Current I/O (input/output) subsystems supported by versions of P.C. Control include

- Analog Devices uMAC 4000 series intelligent I/O subsystem
- Toledo 8132 weigh scales
- Atlantis Scientific's CTC100 digital counter/timer board
- Digiboard COM8 multiple serial communications board
- Stargate ACL-8 intelligent serial communications board
- Data Translation DT2801 series ADC/DAC/Digital boards, supports upto 4 DT2801's
- Data Translation DT2815 DAC board

Atlantis Scientific can write device drivers for other specialized or off-the-shelf acquisition subsystems. In addition, changes can be made to the control program to optimize performance from a particular acquisition board.

3 What's on Your PC Control Diskettes?

The following files are included on your PC Control diskette. Please look at the **READ.ME** file first, if it exists, for update information.

READ.ME If present, it contains the latest information on PC Control changes and corrections.

CONTROL.EXE The main control program. This program runs in the background while the user runs the User Interface Program or any other application software in the foreground. This file may be named something else for your version.

PCC32???.EXE The User Interface Program. This program runs in the foreground, providing the user the ability to modify the process, and to view or log process variables.

DATABUFF.EXE A program that allocates memory for the background CONTROL program.

L3MACROS.EXE A program to load in macros from appropriate files.

PCC32.BAT This is a batch file that activates all of the above, in the correct sequence.

PCC32.LTG This is a user modifiable file, that specifies process variables for logging purposes. See section on Logging for details.

All other files are system files, and cannot be modified:

PCC32.CFG This is a system file. Although this file is in an ASCII format, tampering with it will only lead to improper operation of the menu system.

PCC32.HLP This is a system file. Help information is stored here.

PCC32.ENV This is a system file. The current working environment, such as plotting ranges is saved in this file when exiting the User Interface Program. If missing, defaults are used.

***.BGI and *.CHR** These are graphics support files.

The following files are simply demonstration process files that will be used to demonstrate the use of PC Control.

DEMO.TAG This is a demonstration I/O point file.

DEMO.SCR This is a demonstration process schematic file.

***.MAC** These are demonstration macro files.

***.TMQ** These are demonstration time queue files.

4 Installing PC Control

PC Control is designed to run from the C: drive (hard disk). Installation of the PC Control software begins by making a new directory on your hard disk called **C:\PCC** using the DOS command MKDIR. To do this type after the DOS prompt as follows:

```
C:\>MKDIR PCC
```

Change current path to C:\PCC by typing

```
C:\>CD PCC
```

You need the following subdirectories

C:\PCC\ALARMS—holds Alarm logs

C:\PCC\DATALOG—holds Lotus compatible data files

C:\PCC\ERRORS—holds Error logs

C:\PCC\REPORTS—holds process reports

C:\PCC\VER32—this is where PC Control programs will be copied to.

You create the subdirectories by typing

```
C:\PCC>MKDIR ALARMS
```

`C:\PCC>MKDIR DATALOG`, and so on.

Change to directory PCC\VER32 by typing

```
C:\PCC>CD VER32
```

and copy the contents of all diskettes provided, typing

```
C:\PCC\VER32>COPY A:*.*
```

each time until all the floppy disks are copied. Now the PC Control program is installed onto drive C: in directory C:\PCC\VER32. Note that subdirectories such as DATALOG are not required to start the PCC program since they are simply data storage directories. However, these and all other subdirectories must all be present if all the features of PC Control are to work. You may be confronted with strange error messages if the subdirectories are missing.

The next chapter is a gentle and completely forgiving introduction to PC Control, by way of a special demonstration version of the software. Since no drivers are incorporated in this version, the chances of destroying your plant while learning to use PC Control are pretty slim.

Part II

Getting Started

This chapter is for the first time user of PC Control. A large number of topics are introduced, and all users would do well to read this chapter once. Although references are made below to a demonstration version of PC Control, all instructions apply to your working version of PC Control as well. The reason for including the demonstration version is to have some specific configuration with which to describe features of PC Control. Instead of the demonstration version of the software, you may choose to learn with your working version, by substituting specific references to the demonstration program with the equivalent references of your working versions.

A demonstration program of PC Control is included with your system, and will be used to show the first time user how PC Control is started and to provide the user with some familiarity in the use of PC Control and to provide the user with some elementary process control terminology. The same User Interface program with all its features is used, but the demo program uses a special version of the background program, that has no device drivers, except some simulated analog (or level) input signal fluctuations. Except for the device drivers, the demo version contains all the features of the regular PC Control. As such, the demo version is an ideal learning tool for PC Control, as any devices 'hooked up' to the computer are not affected.

This demonstration is not intended to be all inclusive, but is merely intended to demonstrate some of the more useful features of PC Control. It is recommended that the first time user carefully read the next paragraph before attempting to run PC Control.

1 Invoking the PC Control Program

When first starting PC Control the user simply types the command

```
C:\PCC\VER32\>GO SOMENAME
```

at the DOS prompt. Typing `GO` executes a DOS batch file which activates all the program modules as required. The parameter `SOMENAME` which you specify after `GO` is the name of a tag and screen file to be loaded.

A **tag file** contains the specifications of your 100 input and output points. Stored in a tag file is information such as acquisition device channel numbers, gains, offsets and so on. More explanations will be presented below. The tag file name has an extension `.TAG`.

A **screen file** contains the process schematic diagram. The screen file name has the extension `.SCR`. The first time that PC Control is activated, four programs are executed: the two bf memory resident programs called **DATABUFF** and **CONTROL**, a macro loader program, **L3MACROS** and the User Interface program, **PCC32XXX**. While you can exit from the User Interface, and reload it again, the memory resident programs stay permanently in memory.

Both DATABUFF and CONTROL are only loaded once so that subsequent invocations of GO simply confirm the existence of DATABUFF and CONTROL and ignore the requests, but L3MACROS is executed to load in macros again, and then the User Interface is loaded.

When you exit from the User Interface, and want to reload it again, type:

```
C:\PCC\VER32>GO
```

without any parameter. You may specify a parameter NEWNAME, by typing:

```
C:\PCC\VER32>GO NEWNAME
```

only if you want to replace the previously loaded tag and screen file with NEWNAME. The process that had previously been will be replaced with NEWNAME after a query is issued to confirm that you want to overwrite the running system.

2 Running the Demo Example

Now you are ready to run PC Control with the DEMO process. Simply type:

```
C:\PCC\VER32>GO DEMO
```

Your hard disk will show an enormous amount of activity, as all the files get loaded. In addition, a series of messages will be seen filling the screen. Included will be the sign on messages and names of macro file names being loaded. An error can occur in loading PC Control if a file is missing. Read the error message in that case. If you do not have any errors, all the sign-on messages will be cleared in a few moments.

3 What the Screen Shows

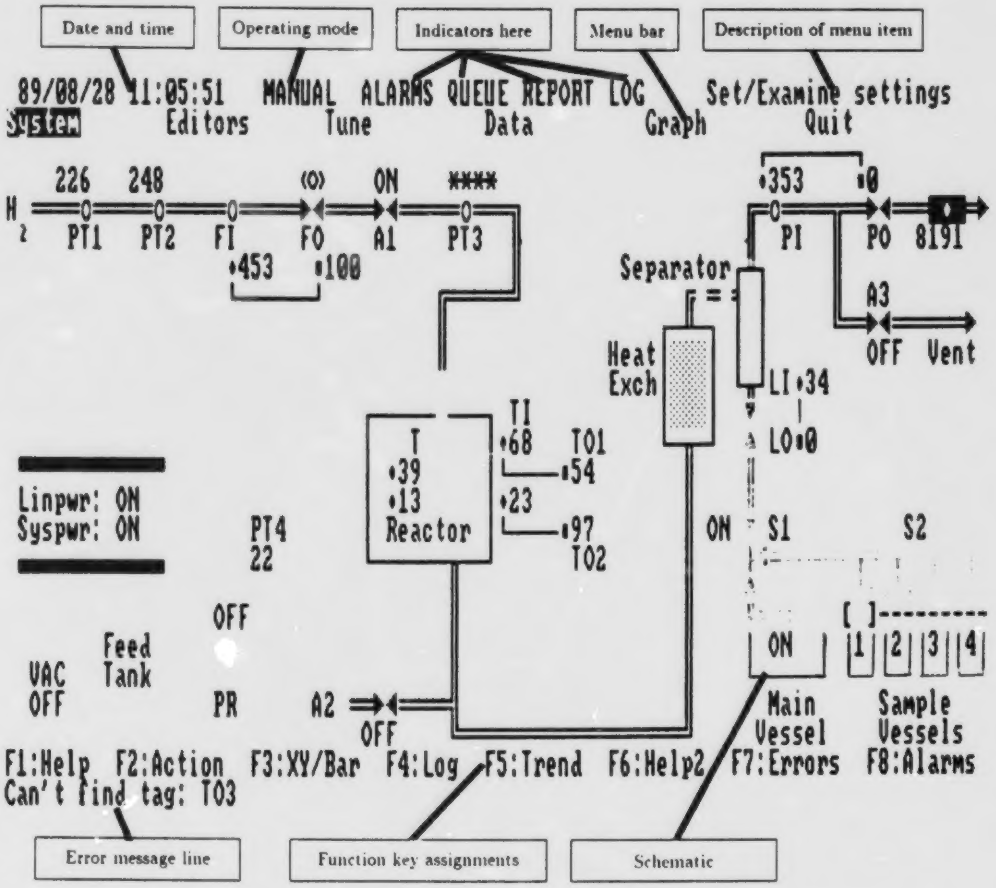
Successful activation of the PC Control system is indicated when a process schematic appears on the screen and a system clock is running at the top left hand corner of the screen, as shown in the figure on page 7. A set of menu items are shown on the second line. Below it is the process diagram. At the bottom are two lines, the lowermost showing the function key operations.

4 What the Demo Process Represents

A process schematic is visible in the middle of the screen, which is that of a chemical reactor for hydrocracking of heavy oils[1]. The rectangle in the centre is the reactor. with lines attached to feed in raw materials (at the top) and outputting reacted products (at the bottom). The product is sent through a heat exchanger, followed by a product separator before being collected into one of several sample containers at the bottom right.

---

[1] Note that the demonstration process is taken from a system using an Analog Devices uMAC4000 I/O subsystem, and an Atlantis Scientific CTC digital I/O board. Although the device drivers are absent from the demo version of PC Control, mention is made of these hardware devices in certain places of the software, which you can disregard.



89/08/28 11:05:51 MANUAL ALARMS QUEUE REPORT LOG Set/Examine settings

System Editors Tune Data Graph Quit

F1:Help F2:Action F3:XY/Bar F4:Log F5:Trend F6:Help2 F7:Errors F8:Alarms

Can't find tag: T03

Date and time
Operating mode
Indicators here
Menu bar
Description of menu item
226
248
(O)
ON
****
H_2
PT1
PT2
FI
FO
A1
PT3
•453
•100
•353
•0
PI
PO
8191
Separator
A3
OFF
Vent
Heat Exch
LI •34
LO •0
TI
•68
T01
•54
•23
•97
T02
T
•39
•13
Reactor
Linpwr: ON
Syspwr: ON
PT4
22
ON
S1
S2
OFF
Feed Tank
VAC OFF
PR
A2
OFF
ON
Main Vessel
1
2
3
4
Sample Vessels
Error message line
Function key assignments
Schematic

Figure 1: Layout of PC Control screen

5 What the Symbols and Numbers Mean

Data values for input (or output) points, or tags can be seen adjacent to the transducers (or actuators). These values can be temperature, or the on-off state of a pump, or the percent output of a heater setting. As you will see in the next section, there are four types of tags: State Inputs (SI), State Outputs (SO), Level Inputs (LI) and Level Outputs (LO), each having different data value types.

The tags have short mnemonic names, such as PT1 (for hydrogen inlet pressure transducer), shown on the schematic at the top left. It is with these mnemonic names that an I/O point can be referred, as will be seen shortly.

6 What are Tags?

Every I/O point has a lot of information to completely describe it. This includes the type of device or board it is using, the conversion gain and offset, the units of measurement, and so on. For example, a thermocouple input may be connected to board X, at channel 3, and has gain 1.0 and offset 0.3, and is designated TI25 (a mnemonic, standing for thermocouple input #25), etc.

All of this information is organized into what is called a tag. Tags are simply data structures containing all the information that makes each I/O point unique. Information such as the I/O point data value, whether the I/O point is a state or level variable, whether the I/O point is an input or an output, the I/O point name, etc is all stored in the tag's structure. Therefore, each I/O point is completely defined by a tag.

There are four kinds of tags: two input types where data values are **input** to computer from a measuring device, and two output types where a data value is **output** from computer to an output device).

The two input and output types are further subdivided into either level (analog) types, having a range of data values, or state (digital) types, having just an ON or OFF state. Examples in the DEMO are shown in the table below:

Tag Type	Tag Number	Tag Name	Description	Units
Level Input (LI)	33	FI	Gas flow rate	ml/min
Level Output (LO)	37	FO	Flow output controller	(0..100%)
State Input (SI)	11	SYSPWR	Line power feedback	(On or Off)
State Output (SO)	9	LINPWR	Line power switch	(On or Off)

7 How the Menu Works

The top left corner of the screen is the current date/time. Next to the system clock you will see a message `OFF`. This is one of several **indicators** that remind us of the modes or functions that are currently chosen. The next line from the top is the menu bar.

If you have not yet pressed any of the cursor arrow keys, you will notice that the second line from the top of the screen is the **root menu** or main menu, with one item highlighted. The menu items are presented and selected just as in the Lotus©123 program, by the use of the <left> and <right> cursor keys and <return> key. One menu item usually leads to another detailed menu. To return to the previous menu, press the <esc> key. A short phrase at the top right of the screen is an explanation of the highlighted menu item.

All functions are grouped in a logical fashion. The root menu items, (`System`, `Editors`, `Tune`, `Data`, `Graph`, `Quit`) represent major groupings of functions, described briefly below:

`System` menu contains functions that apply to the entire system-for example, this is where you go to turn on the alarms

`Editors` menu allows access to the tags, screen, time queue and macro file editors

`Tune` menu contains functions which include those that alter PID control parameters

`Data` menu provides access to the data acquisition parameters, such as file logging interval

`Graph` menu provides functions to all the graphing facilities, such as scale range setting

`Quit` menu item has no submenus - it is the function

8 Purpose of esc key

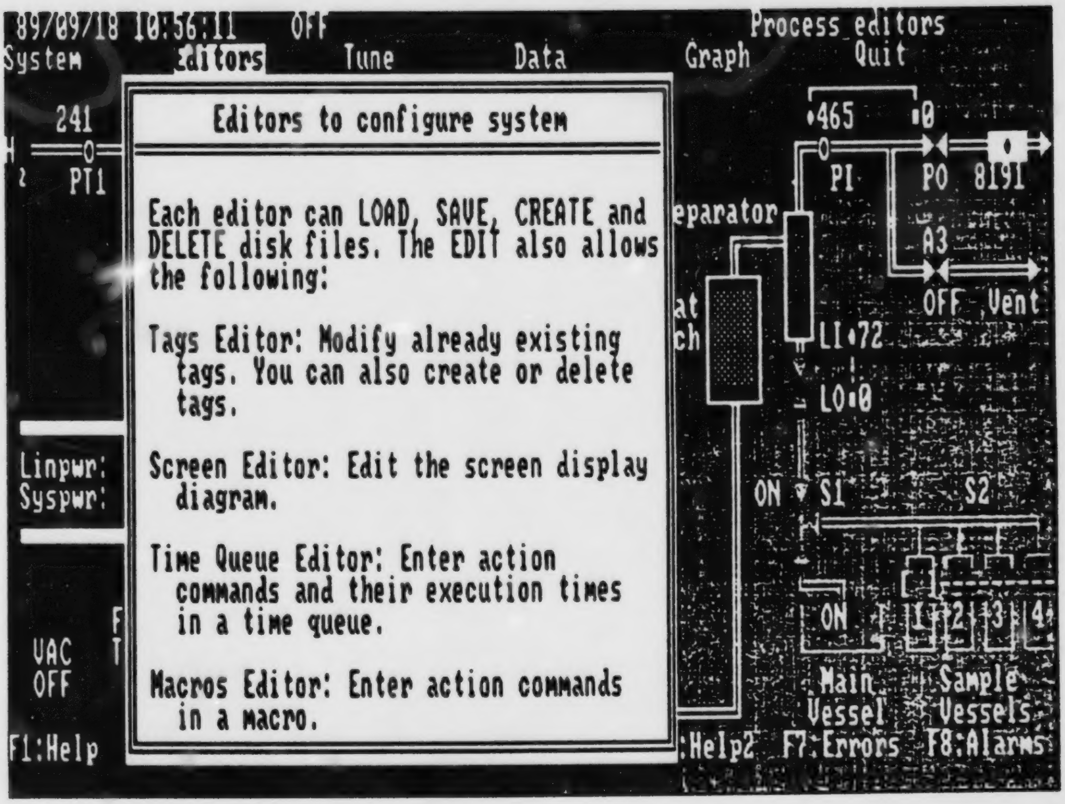
As with the spread sheet program, Lotus 123, the <esc> key allows the user to undo all selections and move back up the menu tree as far as the root menu. The <esc> key also works while entering data from the keyboard in response to PC Control prompts: If the user decides that changing a tag name or altering a plotting limit or whatever was not required, then entering <esc> will cancel the prompt. Thus, pressing the <esc> key allows us to change our mind about a menu selection we just made.

9 How to Quit PC Control

By choosing the Quit menu item from the root menu, you will exit the User Interface. To get back into the User Interface again, simply type GO at the DOS prompt.

10 How to Obtain Help

By pressing <F1>, a help message is presented in a drop down window, on the topic of the highlighted menu item. This is known as context sensitive help. An example screen is shown in the figure on page 10.


89/09/18 10:56:11 OFF
Process editors
System
Editors
Tune
Data
Graph
Quit
241
PT1
465
0
PI
PO
8191
eparator
A3
OFF
Vent
LI 72
LO 0
ON
S1
S2
Linpwr:
Syspwr:
VAC
OFF
ON
1
2
3
4
Main Vessel
Sample Vessels
F1:Help
:Help2
F7:Errors
F8:Alarms

Editors to configure system

Each editor can LOAD, SAVE, CREATE and DELETE disk files. The EDIT also allows the following:

Tags Editor: Modify already existing tags. You can also create or delete tags.

Screen Editor: Edit the screen display diagram.

Time Queue Editor: Enter action commands and their execution times in a time queue.

Macros Editor: Enter action commands in a macro.

Figure 2: Context Sensitive Help Activated

11 Exploring the Menus

The following is a useful excercise for the first time user. The objective is to obtain summary information on various tags. The excercise illustrates the use of the menu structure, while learning more about tags.

We begin at the root menu. If you do not see (`System, Editors, Tune, Data, Graph, Quit`) you can return to this menu by 'climbing' up the menu tree by repeatedly pressing the <esc> key.

Note that the word `System` is highlighted. Press the right arrow key once, and see that the highlighted word changes to `Editors`. By pressing the right and left arrow keys the user can highlight any desired menu item. Notice also that as the arrow keys are pressed that a message at the top right corner of the screen changes. This message is simply a more detailed description of the function performed by each menu item.

Now highlight the word `Editors` and press <return>. Note that the menu line has changed to the items (`Tags, Screen, Time Q, Macros`). These are the four file editors available in PC Control. Next, ensure that `Tags` is highlighted and press <return>.

The menu line should now display the menu selections (`Edit, Load, Save, Create, Delete`). In fact all the editors will present you with a similar menu selection at this level. The selections allow you to make operations on files, i.e. editing, loading, saving, etc.

Ensure that `Edit` is highlighted, then press <return> again. We will edit the current file, the one that is already loaded, DEMO.TAG in our case.

The next menu line is (`Change, New, Delete`). These are the functions you can perform on *individual* tags, being one of 100 tags of the current tag file.

Ensure that `Change` is highlighted and press <return>. Now the menu line will be filled with the tag names. A colon ':' separates the tag number (ranging from 0 to 99) from the tag (mnemonic) name. Move the highlighted item right and left, and notice that every tag can be accessed. At the same time, note that the menu item description at the top right of the screen indicates the tag type (State or Level), whether the tag is an input or output (I/P or O/P) and a more detailed tag description. Now pick `47:KICKER` tag, an example of a state output type.

You will now be presented with this menu: (`Show, Name, Description, Display, Alarms, H/W, Data Cnvrt`). The `Show` function displays a summary of the current tag information while all the other functions allow the user to modify specific components of the tag. Usually, tags are configured for good, and you do not often want to do change their configuration. Aspects of the tags that do need more frequent change, such as alarm trigger settings, usually have a simpler way of changing them—via **action commands**, as you will discover.

We do not want to make configuration changes yet. Pick instead the `Show` item and press <return>. A drop down window presents some information about the tag in question. This is a useful feature in that it allows you to have a summary of a particular tag, that you can refer to whenever you are making configuration changes.

Pressing <return> again should bring the `Show` menu back into view. Pressing <esc> will bring you back to the `Change` menu item.

Try selecting a 'State I/P' type tag now and view its contents on your own. An example of this is 11:SYSPWR.

Now move the cursor to the right until you find `33:FI`, a 'Level I/P' tag. Select it and then do a show. The user will see more information is displayed for the level variables than for the State variables. Note that a transformation of the ADC (analog to digital converter) data to engineering units is obtained by programmable gain and offsets. Note also that detailed alarm information is displayed.

Four alarms levels are offered, namely HiHi, Hi, Lo and LoLo. The alarm setting for each level is shown, along with the corrective or emergency action command to perform should that alarm be set off. The command language is explained in the next section. If the value of FI becomes greater than Hi or HiHi or becomes less than the value of Lo or LoLo then an alarm occurs[2]. An alarm situation for this tag is indicated when its data value on the process screen begins to flash, and then the programmed corrective actions are performed.

This completes our brief tour down one branch of the root menu. Other items of the root menu will be explored in the next few sections, but the user should now have a good feeling of the menu hierarchy.

12 Using the Command language

Our objective in this section is to enable the global alarms[3], or if you will, to turn on the master switch to alarms, by the use of a command. We will also do this via the menu system first, and then show the equivalent command that you can type into the system.

Alarm Toggle via Menu

We begin at the root menu. If you do not see (`System, Editors, Tune, Data, Graph, Quit`) you can return to this menu by climbing up the menu tree by repeatedly pressing the <esc> key. Pick item `System` this time, and press <return>. The next menu list is (`Mode, Toggles, Check, Info, Version`). The item `Toggles` contains the on/off switches for, amongst other functions, the alarms. So pick `Toggles` and then choose `Alarms` from the list of toggles, and press <return>. You are now shown a prompt

```
Alarms are OFF : Toggle [Y]/N ? :
```

This prompt is shown at the bottom of the screen. *Whenever input is requested from the user, a prompt is shown at the bottom of the screen.* The yes selection is default, as indicated by the square brackets surrounding that selection. Enter a 'Y' followed by <return>. The alarms will now be turned on, as shown by the `ALARMS` indicator on the top of the screen. The `ALARMS` indicator is one of several that appear at the top of the screen, reminding us of the modes and function that are chosen. You may or may see an alarm event occurring if a certain tag's value is flashing upon the screen. You can disregard that for our excercise.

---

[2] The HiHi alarm limit is placed higher than or equal to the Hi limit. Similarly, the LoLo alarm limit is set at lower than or equal to the Lo alarm limit.

[3] Note that in addition to the global alarm toggle, each input tag has its own alarm toggle. The global alarm toggle supersedes the individual tag alarm toggles.

Try turning the alarms off now by using the **Alarms** menu item again and pressing <return>. This time you are prompted with

```
Alarms are ON : Toggle [Y]/N ? :
```

Just press <return> to accept the default yes response. When you have turned the alarms off, the ALARMS indicator will go out.

Alarm Toggle via Action Command

Now to the equivalent typed commands to toggle alarms. The two commands are ALARMS ON ALARMS OFF and the way to enter a command into the system is as follows: Press <F2> function key.

You will notice another prompt at the bottom of the screen

```
Enter ACTION Command:  [ ] :
```

Type in

`ALARMS ON <return>`.

After a short pause, the alarms will be turned on, as evidenced by the ALARMS indicator coming on. To switch the alarms off, type in the following:

<F2> ALARMS OFF <return>.

The alarms will be shut off after a short pause. So we see that the action commands accomplish the same thing as the menu function. They provide a short cut to achieving an action without having to go through the menu structure. Action commands have a more important function: they are used in time queues and in alarm actions, allowing control changes that are made without operator intervention.

13 The Function Keys

You have already come across the <F1> and <F2> function keys: The F1 key provides context sensitive help, while the <F2> function key allows the entry of an action command. <F2> allows the user to enter many of the actions without having to find them via the menu structure. In essence, the function keys allow access to useful features no matter where in the menu you are.

For a list of the other function keys, look at the line at the bottom of the screen. Two graphing features provided by PC Control are the real time XY or bar graphs, obtained by pressing <F3>, and historical trend plots, obtained by pressing <F5>. These functions will be described in the next section.

By pressing <F4>, a log file (conforming to the Lotus©number import format) is created. Normally, data log files are generated automatically at the preset **logging interval**, provided the automatic logging feature is toggled on. The <F4> key is a supplement to this—it is an additional way to generate a data log file at the instant that key is pressed, regardless of the automatic logging toggle.

The <F6> function key is a way of obtaining help on topics not available through <F1>. Remember that the <F1> key provides help on menu items.

PC Control maintains two running logs

1. **Error Log**—A log of system errors, with each error date and time stamped.
2. **Alarm Log**—A log of alarm events, with each alarm event date and time stamped.

The error and alarm logs can be viewed on the screen by pressing <F7> and <F8> respectively. They are ASCII files, kept in their own directories. Their contents can be viewed with any editor, and provide crucial information when tracking down errors or the progress of unattended plant operation.

14 Obtaining Graphs

Before we can display the graphics functions, we will change the state of our demo system from Off to **Monitor mode**. *When the system is changed from Off to Monitor mode, it begins reading in data from the level input (LI) and state input (SI) devices attached to the computer*. If you are running the demo version of the software, all LI tags will begin updating with simulated data. But if you are running your working version, all LI and SI tags will be read in from the data acquisition boards.

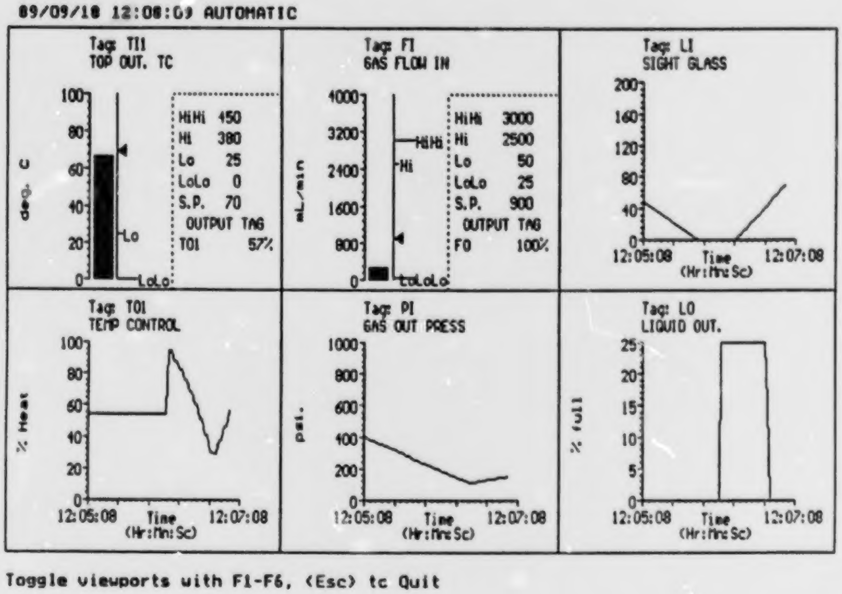
Beginning with the root menu, proceed as follows: Pick items `System` then `Mode`. You will see the following prompt:

```
Process is OFF : set to MONITOR [Y]/N ? :
```

Enter a `Y` and <return>. Notice the mode indicator at the top of the screen changes from `OFF` to `MONITOR`. In the demo version of PC Control, the numerical values of the LI tags will begin fluctuating; these have been programmed to follow random ramps, for illustration purposes.

XY/Bar Real-Time Plots

To see a real-time plot of the values against time by using the XY/Bar function, press <F3> and see what happens. You should see six graphs being plotted in real-time. Watch the graphics display long enough to see the plot scroll left once the maximum time along the X axis is reached. Press <F1> now. The graph at the top left should toggle between an XY format to a Bar graph format every time <F1> is pressed. Similarly for the other 5 plots. Note that the Bar graph both writes the value of alarm limits and set point in an outlined box to the right of the graph and also marks the location of alarm limits and set point along a secondary axis located to the right of the Bar. Pressing <esc> returns the user to the menu where trending was invoked. An example display is shown in the figure on page 15.



```
09/09/18 12:08:09 AUTOMATIC
```

Tag: TI1
TOP OUT. TC
deg. C
100
80
60
40
20
0
Lo
LoLo
HiHi 450
Hi 380
Lo 25
LoLo 0
S.P. 70
OUTPUT TAG
TO1 57%
Tag: FI
GAS FLOW IN
mL/min
4000
3200
2400
1600
800
0
HiHi
Hi
LoLo
HiHi 3000
Hi 2500
Lo 50
LoLo 25
S.P. 900
OUTPUT TAG
FO 100%
Tag: LI
SIGHT GLASS
200
160
120
80
40
0
12:05:08
Time
(Hr:Mn:Sc)
12:07:08
Tag: TO1
TEMP CONTROL
% Heat
100
80
60
40
20
0
12:05:08
Time
(Hr:Mn:Sc)
12:07:08
Tag: PI
GAS OUT PRESS
psi.
1000
800
600
400
200
0
12:05:08
Time
(Hr:Mn:Sc)
12:07:08
Tag: LO
LIQUID OUT.
% full
25
20
15
10
5
0
12:05:08
Time
(Hr:Mn:Sc)
12:07:08

```
Toggle viewports with F1-F6, <Esc> to Quit
```

Figure 3: Real Time Plot of Demo Process

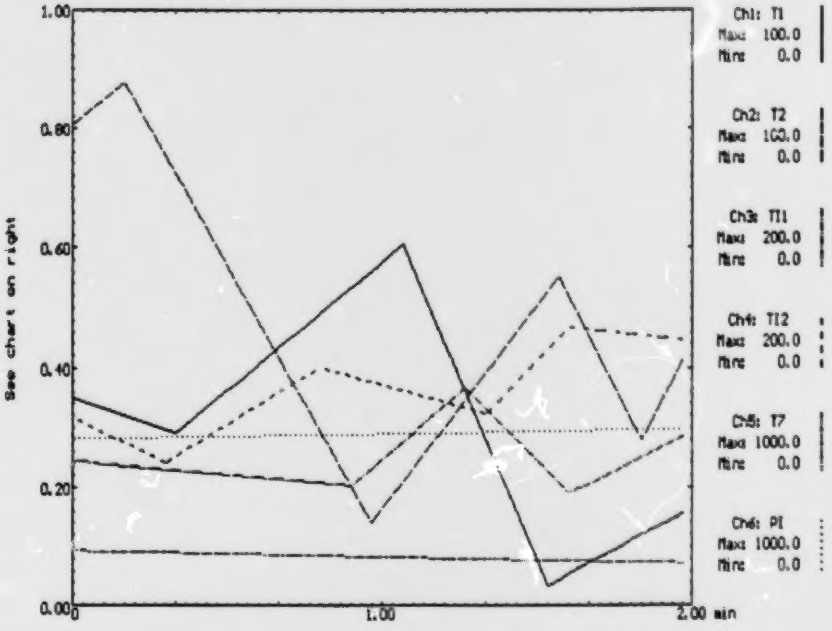


Figure 4: Historical Trend Plot

Historical Trend Plots

To observe a historical trend plot of 6 tags on a common axis, press <F5>. On a colour monitor, each of the 6 trends will be a different colour, while on a monochrome monitor, different line patterns are used. A key on the right of the screen shows the assignment of each line style to a tag. A sample graph is shown on page 15. To exit from the plot, press <esc>.

Editing of trend plotting limits, tags to be plotted, can all be done from the `Graph` menu, reached from the root menu. The plotting interval can be changed from the `Data` menu. Look at the chapter 11 for more details.

15 Screen or Schematic Editor

We introduce the screen editor by doing the following short exercise. Move back to the root or main menu by repeatedly pressing <esc>. Select `Editors` and then `Screen`. Screen files are named with an extension `.SCR`. Ours is `DEMO.SCR`, and is already loaded. You can choose to load an alternate screen, by using the `Load` menu item. This way, you can have as many different screens (or views) of your process as you wish.

As an example, we have provided an alternate view of the reactor temperatures in `DEMOT.SCR`. To view this, pick `Load` menu item, then `DEMOT` item. You will notice a new schematic, one detailing just the reactor temperatures. Also note that the temperatures are now presented with a new format, showing an extra digit of precision. You can return back to the `DEMO.SCR` screen simply by reloading that screen again. Editing a current screen is achieved via the `Edit` menu. If you have altered your screen layout, and wish to save your work, you return back to this menu level, and pick the `Save` menu item.

16 How to Change Level and State Output Tag Values

Our first objective is to change, via the menu system and via an action command, the level output tag, TO1 from its current output value to a new value of 80 (percent of maximum heater output).

First locate TO1, which is the upper of the two heaters seen in the middle of the screen. To make certain we can see the change, note the current heater output value. Type in the following

```
<F2> $TO1 DATA 80 <return>
```

and in a little while, the heater output value changes to 80, as desired. *The $ in front of a tag name in an action command signifies a reference to a tag.*

The same action can be achieved through menus by using item `Tune` from the root menu, then choosing `L.O.` then `41:TO1` then `Data`. When you choose the last item, you are prompted with

```
Enter new data value for TO1 :[ 54.0 ] :
```

at the bottom of your screen. Shown in the square brackets is the current value of TO1. You may have a different value. Type in `80 <return>`, and the heater output will be changed as in the command action.

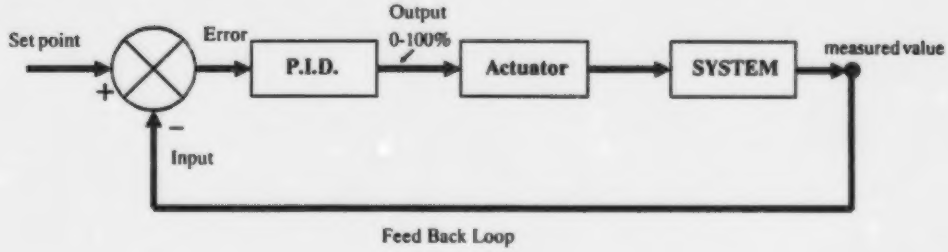


Figure 5: PID Control Loop

Quite often, we wish to change a whole number of LO tags at once. In section 20, you will see how you can combine several command actions together into a macro, simplifying your task.

To change a state output tag from its OFF to ON state, or vice versa, we use another choice from the **Tune** menu item: `S.O.`. After you pick this menu item, you choose a state output tag from the list. On pressing return after picking an S.O. type tag, a prompt appears that allows you to flip the state from it's current state. You can also use an equivalent action command:

`$TAGNAME ON` or

`$TAGNAME OFF`

to achieve the same thing.

17 What is PID Control?

PC Control offers PID (proportional plus integral plus derivative) control loop mechanism, and this section will introduce the concepts behind PID control.

In a control loop, we have three important parameters: the **input**, the **output** and the **set point**, as illustrated by Figure 5.

With reference to the demo process, suppose we want to maintain the temperature of the outside of the chemical reactor, TI1 to a certain value automatically. This value is our set point temperature. We can measure the actual temperature with a thermocouple, TI1. This is our input or controlled variable. An electric heater around the reactor wall, TO1, heats the vessel. This is our output variable that we can control. This example is shown by Figure 6. When the heater is on, the reactor temperatures rises. When the heater is off, heat is lost via conduction, convection, and radiation, all of which can be referred to as **loads**, which allow the reactor to cool.

There are many factors causing fluctuations in temperature, such as changes in conduction rate, power variations, and so on. A change in load in our demo process is caused, for example, by a

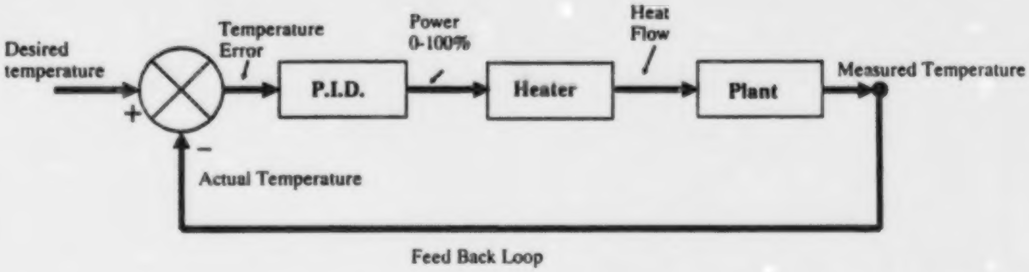


Figure 6: Temperature Control Loop

different rate of heat loss. This may originate from changing the flow rate of products through the reactor.

The computer calculates the difference between the required set point temperature and the measured temperature. This is called the **error** value. *Proportional control changes the output proportionally to the error signal.* This maintains a constant temperature, provided no load changes have occurred. The corrections can be thought of as an AC (alternating current) signal that hovers around the DC (direct current) offset of the output. *Exact correction is provided through proportional control only for one particular load.* For all other loads, there still exists a deviation or offset error. It is the purpose of the integral action to correct this offset error.

The integral action part contributes to the output continuously, for as long as an error signal exits. This corrects for changes in load that occur while trying to maintain the temperature constant, by changing the DC level of the output. *The integral action responds both to the magnitude of the error signal, and the time duration of it.*

In proportional plus derivative action (rate action), the rate of change of the error signal governs the output variable. Hence, the faster the temperature changes, the higher is the contribution from the derivative action. The derivative term is, in a sense, an anticipatory change since it is based on the current rate of change of error, and assumes that rate will stay the same. The main disadvantage to derivative action is that if you have a noisy error signal, even if it is close to zero, the overall control action can have unnecessarily large fluctuations. Quite often, control systems consist of just proportional and integral control.

Since the demo process does not respond to control changes, PID control cannot be effectively demonstrated in this section. You can turn the demo process to automatic, anyhow, to examine other aspects.

PID control parameters are set via the **Tune** menu item of the root menu. The PID parameters, such as set point and input variable are all specified in the associated output tag. This means you pick the `L.O.` menu item, after the **Tune** menu item. Then, pick the output tag, for example `41:T01`, the heater in the demo process. Within this tag are specified control parameters such as the P, I and D gain constants, the input tag, which happens to be TI1, and other PID control information. To modify any of these parameters, pick **Loop1**[4].

PC Control has a large number of action commands to achieve control parameter changes, such as assigning set points, and set point increments. This allows great flexibility in controlling equipment when these action commands are used in a time queue.

18 Set Point Ramping

For some processes a sudden change in the process operating conditions may be harmful to the process, for example changing the speed of a large heavy duty engine from one value to another by a sudden step change of the throttle position. The speed of the engine is normally controlled by monitoring it (the input variable), and adjusting the throttle (the output variable) to achieve the required speed (set point), using PID control. In a chemical reactor, the temperatures in the reactor are brought up slowly to their final values. For such applications, the set point can be changed gradually from the current value to the new target value, by automatically ramping the set point.

PC Control provides a facility for automatic set point ramping. By specifying the **target set point** and the **ramp period**, the software manipulates the set point from its **current set point** to final in a linear fashion. A number of action commands are provided to program set point ramps. The ramping parameters can also be set via the **Tune** menu from the root menu.

19 Alarm System

PC Control incorporates an alarm processing capability for LI (level input) and SI (state input) tags. Alarm trigger points can be examined for LI or SI tags by using the **Show** menu item of the tag editor. As an example, let us examine the alarm information for the tag named FI in the demo process, the gas flow rate into the reactor.

From the root menu, pick in succession `Editors` then `Tags` then `Edit` then Change, then `33:FI` and finally **Show**.

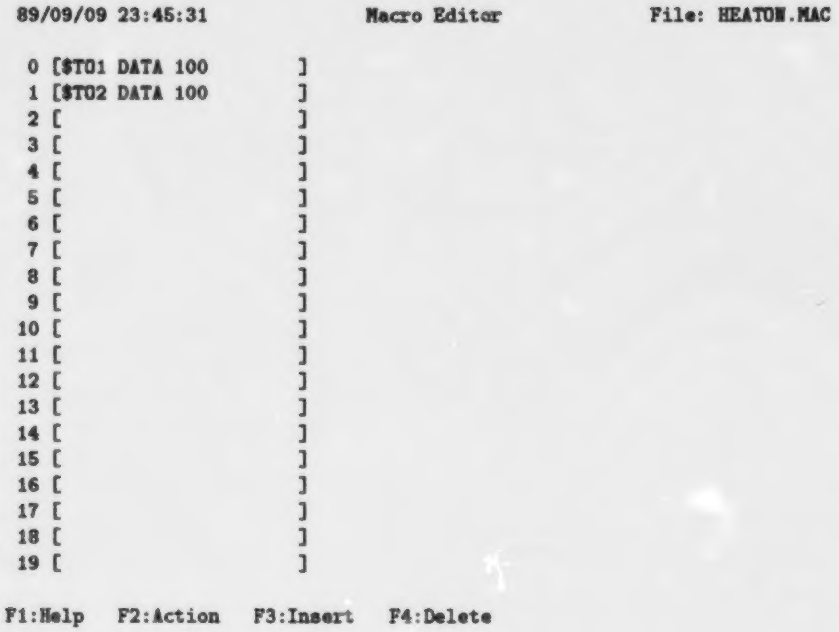
In the case of level input tags such as 33:FI, four alarms levels are offered, namely HiHi, Hi, Lo and LoLo. If the value of the tag becomes greater than Hi or HiHi or becomes less than the value of Lo or LoLo then an alarm occurs. The HiHi alarm limit is placed higher than or equal to the Hi limit. Similarly, the LoLo alarm limit is set at lower than or equal to the Lo alarm limit. An alarm situation for a LI tag is indicated when its data value on the process screen begins to flash. In addition, the programmed corrective actions are executed.

For SI tags, an alarm can be programmed to occur when the tag reads a logic high or low reading.

The alarm setting and action commands for each alarm type is entered via the **Alarms** menu item. To get to this menu item, press <esc> to leave Show, and then pick **Alarms**. Alarms trigger

---

[4] Note that only loop 1 of the three possible loops should be used. Loop 1 must have 100% weighting. No support for loops 2 and 3 is provided, and future versions of PC Control will remove multiple loop option.



```
89/09/09 23:45:31              Macro Editor              File: HEATON.MAC

  0 [$T01 DATA 100          ]
  1 [$T02 DATA 100          ]
  2 [                       ]
  3 [                       ]
  4 [                       ]
  5 [                       ]
  6 [                       ]
  7 [                       ]
  8 [                       ]
  9 [                       ]
 10 [                       ]
 11 [                       ]
 12 [                       ]
 13 [                       ]
 14 [                       ]
 15 [                       ]
 16 [                       ]
 17 [                       ]
 18 [                       ]
 19 [                       ]

F1:Help   F2:Action   F3:Insert   F4:Delete
```

Figure 7: Macro Editor with File:`HEATON.MAC`

points can also be changed via command actions, making it convenient to define different process scenarios with the same plant equipment.

20 Macros, an Example

Macros are a set of action commands to be performed one after another. As an example, a heater is shut off, a valve is opened, and another is closed. When such a series of actions need to be performed often, then the action commands need only be written once and kept in a macro. Each time you want to perform the series of actions, represented by many action commands, you simply replace them by an action command to execute the macro. A macro is analagous to a subroutine in a computer program. Several corrective actions for an alarm condition can be collected into a macro.

The advantages of using a macro include

- condensing a whole series of command actions into one, thereby reducing operator error during command entry
- reducing time needed to test frequently entered series of action commands
- a descriptive name can be given to a macro, eg FEED_OFF

The macro is given an 8 character name, and is saved in a file with extension `.MAC`. PC Control incorporates a macro editor, with automatic syntax checking of action commands, as your enter them.

All macros are loaded into a special area of memory, called the **macro buffer**, *every time PC Control is invoked. The macro editor does not directly alter anything in the macro buffer. Instead, it allows the user to modify or develop new macros that only reside on disk files. If macro files are changed, PC Control must be exited and re-run to load in the changed macros into the macro buffer.*

To illustrate macro use, we will examine one that has already been written, and then execute it. Press <esc> key until you see the root menu. Then pick the following: `Editors` then `Macros` then `Load`. Pick the macro file `HEATOFF` to load.

Now the contents of the file are loaded into the editor. To examine the contents, pick the `Edit` item. Figure 7 shows an example of the screen display. Each action command is placed on a line. The command actions simply turn heaters TO1 and TO2 to zero percent heat output. To execute the macro (to perform the action commands contained within it), begin by pressing <esc> once to leave the editor. Keep your eye on the current TO1 and TO2 heater output values, to the right of the reactor. Then type in the action command to execute our macro as follows:

```
<F2> EMACRO HEATOFF <return>
```

After a brief pause, both heater values will be sent to zero percent output. Another macro, `HEATON` has been designed to turn the heaters to maximum. Type in:

```
<F2> EMACRO HEATON <return>.
```

To examine what actions make up `HEATON.MAC`, load that file and pick `Edit` to view it as before.

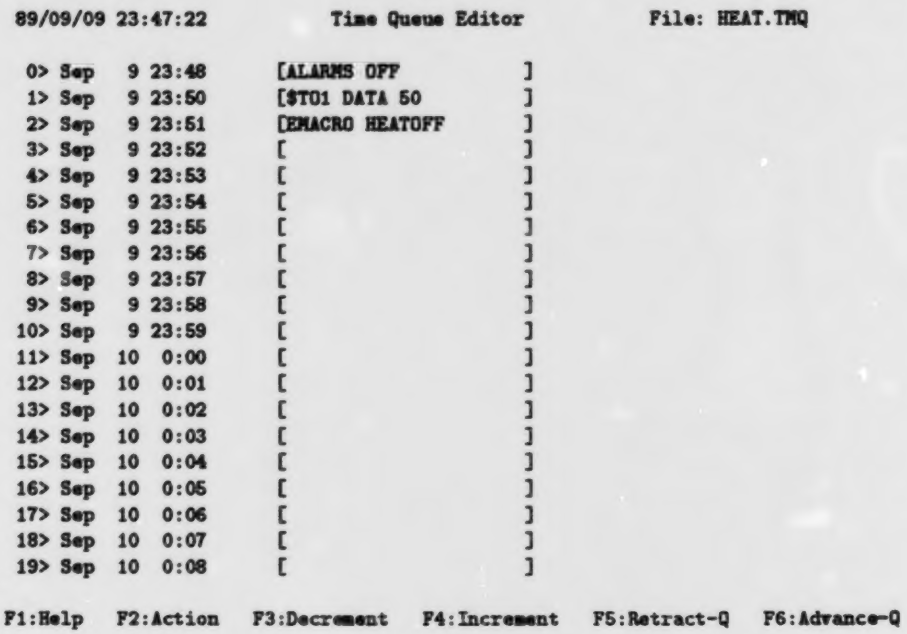
21 Time Queues, an Example

The purpose of a time queue is to execute a command action at a given time. With this capability, unattended control of a process can be achieved. We will learn the principles behind a time queue as we examine a simple time queue and run it.

Time queues are kept in files having extension `.TMQ`, and are loaded into PC Control as needed. A time queue file editor has been incorporated into PC Control. Writing a time queue program is easy with the editor. It is also a convenient way of examining a previously written time queue program. To access it begin at the root menu, and pick `Editors`, followed by `Time Q`, followed by `Load`. Pick file `HEAT`. To look at what that file has, now pick `Edit`. Figure 8 shows an example of the screen display.

Shown on your screen will be the current date and time at the top left corner as usual, and three command actions, set to execute on the date and time shown at the left of these actions. This is a queue of actions.

Note that the first action, to turn the alarms off was set for 1 minute past the time the time queue was loaded. This time may have already passed as you read this sentence! Don't panic. The second action, to turn heater TO1 to 50, is set for 2 minutes past the first. The third action, to execute macro HEATOFF (see previous section), is set for 1 minute past the second action. If we



```
89/09/09 23:47:22          Time Queue Editor          File: HEAT.TMQ

  0> Sep  9 23:48      [ALARMS OFF          ]
  1> Sep  9 23:50      [$TO1 DATA 50        ]
  2> Sep  9 23:51      [EMACRO HEATOFF      ]
  3> Sep  9 23:52      [                    ]
  4> Sep  9 23:53      [                    ]
  5> Sep  9 23:54      [                    ]
  6> Sep  9 23:55      [                    ]
  7> Sep  9 23:56      [                    ]
  8> Sep  9 23:57      [                    ]
  9> Sep  9 23:58      [                    ]
 10> Sep  9 23:59      [                    ]
 11> Sep 10  0:00      [                    ]
 12> Sep 10  0:01      [                    ]
 13> Sep 10  0:02      [                    ]
 14> Sep 10  0:03      [                    ]
 15> Sep 10  0:04      [                    ]
 16> Sep 10  0:05      [                    ]
 17> Sep 10  0:06      [                    ]
 18> Sep 10  0:07      [                    ]
 19> Sep 10  0:08      [                    ]

F1:Help  F2:Action  F3:Decrement  F4:Increment  F5:Retract-Q  F6:Advance-Q
```

Figure 8: Time Queue Editor with File:`HEAT.TMQ`

were to execute this time queue, we would:

1. turn alarms off after one minute of starting
2. send heater TO1 to 50% heater output after another 2 minutes
3. turn heaters TO1 and TO2 to 0% after another minute.

A time queue file contains a series of time offsets and accompanying action commands. When a time queue is loaded, the program (by adding the current time to the time offsets), arrives at a time table of future command actions to perform. So, what matters is:

1. the relative time difference between queue items or time offsets and
2. the time at which the queue is loaded.

Since the current time is probably past some of the set times in our time queue, we will execute this time queue by reloading it. We will do this not through the menu, which is time consuming, but by an equivalent action command.

Press <esc> to exit the time queue editor. We will turn on the time queue first. A time queue will not operate until this is done. As well, the system must be in monitor mode, shown by the indicator `MONITOR` atop the screen. If your system is not in Monitor mode, refer back to Section 14. Type in

```
<F2> QUEUE ON <return>
```

and watch for another indicator, **QUEUE** on top of the screen to confirm this action. Next we will load the time queue HEAT.TMQ via an action command, so that we can keep attention on the process screen. Type in

```
<F2> QLOAD HEAT <return>
```

As soon as the time queue is loaded, all the time offsets from current time are applied. This means our first action, (ALARM OFF) will occur at the top of the minute, as the seconds reach zero. Watch for the actions to occur as programmed. If you note the current time, you can predict when every action will occur. After you have satisfied yourself of this operation, it is advisable to turn the queue off, by typing in the action

```
<F2> QUEUE OFF <return>.
```

Note how the use of the Time Queue allows programming of process conditions to occur at specific times. Herein lies the real power of PC Control; the ability to change process conditions at specific times as well as the ability to automatically maintain process operating conditions. This combination greatly reduces the time required to supervise any process operation.

Part III

Reference Section

1 System Overview

1.1 Introduction

Before entering into a detailed description of PC Control operation, it may be instructive to first examine what PC Control allows the user to do on a conceptual level. This chapter provides the user an understanding of the scope of PC Control and how the different components of the hardware and software work together.

PC Control is a microprocessor based process controller designed to meet the needs of small scale processes such as pilot plants and laboratory installations. PC Control is actually two (main) separate programs: 1) a control program, CONTROL, which runs in the background and 2) a user interface, PCC32XXX, where the XXX are some letters used for your version, that runs in the foreground. There are other subsidiary programs, namely the macro loader, L3MACROS, whose purpose is to load all macro files into the macro buffer, and DATABUFF, a program that makes the computer set aside some memory for data storage.

The background control program is a memory-resident program that is always present while the user interface is not memory resident and terminates upon exit. The control program is in an interrupt driven background state, which means that it can begin executing by interrupting whatever foreground program is running. This means that the background and any foreground program execute simultaneously, sharing computer time in alternate time slots. The user interface can be run in the foreground initially to set up the entire control strategy, which is recorded in the background program, and then it can be deactivated in order to run other application programs. Thus, PC Control can be performing the programmed control strategy while the user is running other software such a text or word processor to examine data files, or Lotus©123 to view and process the data files.

PC Control has a very broad range of application because it has been designed with flexibility in mind. Application of PC Control has been successfully performed in controlling laboratory sized chemical reactor vessels in particular. In addition to the chemical processing applications, it has been installed to control and monitor a diesel research engine. In fact there is virtually no limit to the possible applications of PC Control, with the only constraint being that the total number of I/O points in the proposed process be at most[1] 100.

Automatic process control is the primary function of PC Control. Automatic control involves monitoring a system parameter, feeding this through a PID equation and adjusting output in order to maintain the monitored system parameter at a very constant level. A great deal of flexibility is offered in the software to set up automatic control of system parameters. Currently PC Control supports several I/O devices, as indicated in section 2 of *Getting Started*. And, the addition of new peripheral devices is easily accommodated through the use of linkable drivers.

---

[1] The total number of points for PC Control was chosen as a compromise between having a sufficient number and not too many as to burden the computer with excessive processing. PC Control works satisfactorily even on an (8088 microprocessor based) IBM PC, at sample periods of 0.5 seconds

Some of the more important software capabilities are listed with an indication of how the individual features can be used to customize any installation.

Process Schematic A schematic of the process can be generated using an editor which allows the placement of IBM pseudo graphics characters anywhere on the screen. Attributes of the characters making up the schematic can be selected (ie foreground and background colours etc). In addition, the data value for each I/O point can be located anywhere on the screen with the desired number format.

Programmable Alarms Four alarm levels are available with PC Control. If the I/O point data value exceeds the alarm limits then the output on the process schematic begins to flash and programmable alarm action commands can be executed to obtain corrective or safety measures.

Manual and Automatic Control Output variables, such as heaters, actuators etc, can be set manually or can be controlled automatically via tunable (variable gain) PID control loops. Tuning of PID parameters plus the ability to enter signal calibration coefficients makes a unique description of any process variable possible.

Real Time Trending Graphical displays of up to six process variables is possible. Graphs can be in cartesian or bar format. Note that tuning of control loops becomes greatly enhanced by viewing the trends of the monitored input against that of the output variable.

Real Time Logging and Reporting Automatic report and data log generation is possible at programmable rates and in real time.

Time Queued Actions Action commands can be placed beside the desired time for this action to occur. A list of actions and associated times can be generated to program PC Control for performing actions at a later time (ie. changing speed and load of an engine, reducing the temperature of a chemical reactor vessel, opening/closing valves, etc). This is a powerful feature as it allows for performing a sequence of operations without supervision of the program.

Executive and Menu Oriented User Interface The user interface provides access to all facets of the process control via a Lotus like menu system and an executive feature. Many of the operations that are embedded in the menu structure can be directly performed by an executive command issued by the user from almost any menu level.

Two Help Levels On-line help is available in a context sensitive manner which explains each menu option and in an extended manner which provides general explanations of system features and operation.

Macros Creation and execution of Macros, allow single word execution of longer command sequences. This allows the operator to define custom command words to perform frequently used sequences of commands.

1.2 Theory of Operation

The background program, CONTROL, is written in Microsoft C and assembly language, while the foreground, user interface program called PCC32XXX is written in TURBO PASCAL v5.0. Even though it is a complex system, the user does not have to understand all of it, since every attempt has been made to keep the program easy to use. However, reading the following will help in understanding what goes on in the program.

The background program becomes activated as a result of the timer tick interrupt, numbered 1C hexadecimal. The timer tick is a hardware timed pulse, occurring at approximately 1/18 of a second. Since PC Control operates at a sample period of 0.5 or 2.0 seconds, depending on the version, the first task CONTROL performs is to determine if 0.5 (or 2.0 seconds) have elapsed. If so, then the CONTROL program executes its data acquisition and control section. Otherwise, it gives up control of the computer, and allows the foreground program, be it the User Interface, or Lotus©, or other program to execute. This is why you might observe a pause in you foreground task, when the background task has taken over.

Apart from the program itself, there is a 'reserved' section of memory that maintains a collection of data structures that hold the control strategy. It is set up when you first load PC Control, and remains there until the computer power is shut off. This 'data base' includes:

PC CONTROL'S DATA STRUCTURES

Tag structure	The 100 tags, with their various components
System toggles	Toggles including the alarm toggle, logging toggle, and so on
System settings	Trending interval, logging interval and so on
Macro buffer	This buffer has in it loaded macro files from the disk
Time queue buffer	This buffer has in it a loaded time queue file
Trend buffer	Holds 60 previous values of all 100 tags, used in graphing and logging
Error log	Holds 45 entries of date and time stamped system error messages
Alarm log	Holds 45 entries of date and time stamped alarm events

Invoking PC Control at the DOS prompt for the first time causes DATABUFF to be executed first. This sets up the trend buffer, error log and alarm log data structures in memory. The CONTROL program sets up the other data structures. Both these programs keep an identity signature in a special location, which is used in detection of wether the programs are already loaded or not. When the independent program macro loader, L3MACROS is run, it searches the current directory for files ending with the extension `.MAC` and loads them into the macro buffer memory area. Subsequent invocations of L3MACROS cause the macro files to be loaded into the macro buffer once again, effectively overwriting the previous contents of the macro buffer. When the background program CONTROL is first invoked, the tag structure is empty.

The first job for the User Interface program is to load the contents of a specified **tag file** into the tag structure. It does this by accessing the same tag structure and it also allows the user to manipulate tag information thus altering the control strategy. From its integrate tags editor, the User Interface allows the user to save entire tag structures to disk and also load tag structure data from existing files. The User Interface program has the primary function of viewing or editing the components of the data structure. These components include the tag structure and the time queue. Because the background program must maintain continuous, synchronous operation, no time consuming tasks, such as file loading and saving can be performed from within it. Instead, *the User Interface incorporates all of the file loading and saving functions*. It is for this reason that the User Interface must be kept running if automatic data logging is required. More on this in Section 11.

Note that the background/foreground process control concept is really the only way to achieve constant process control. For instance consider a process that requires constant and precise controlling and the program is accidentally exited. For PC Control this scenario is of no consequence because the CONTROL program cannot be exited but the user interface can. Another advantage to a background, memory resident CONTROL is that the complete control strategy can be programmed into PC Control, including PID control, the alarm processing system, and system changes via time queues, and then the user interface can be terminated. Thus, the computer can be used for process control and, depending on the available memory size and speed of the system, another application program such as Lotus or a word processor can be in use at the same time.

Given that the background and foreground programs function together without 'knowledge' of one another the user is cautioned to carefully note which features are performed by which program. The main functions that are carried out by the *background program* include:

1. Data acquisition and PID or manual control of outputs. Data acquired is placed in the input tags, and control data to be output is obtained from output tags.
2. Storage of data into the trend buffer, which holds upto 60 samples of all 100 tags. This is used for data logging and graphing.
3. Checking input tags for alarm situations, and maintaining an alarm log.
4. Interpretation and execution of action commands, including macros.
5. Maintaining the error log which contains both software and hardware related errors.

User Interface and its Support Files

The foreground User Interface is a very versatile program which provides the user with a tool to send control instructions to the background CONTROL program. In addition to providing access to the background program, PCC32XXX provides graphics support, logging of data to disk, reporting, and visible indications of alarm conditions. The PC Control User Interface program consists of a Lotus 123 style menu system with on context sensitive help facility. All features of the PC Control system may be accessed through the menu structure while some short cuts to frequently used features are accessible through function keys or executive action commands. Both the **menu system** and the **help information** are loaded into the user interface program from two disk files `PCC32.CFG` and `PCC32.HLP` respectively.

Whenever you exit PC Control's User Interface (for the purpose of operating other application software), a number of parameters defining the current configuration, or the **environment** of the system are automatically saved to a disk file, `PCC32.ENV`. These parameters include such things as scale limits for graphics, current screen file name, and so on. If this file is erased then the User

Interface chooses some default settings for you. All the features of the User Interface program are covered in more detail in the following chapters.

2 Operating Modes for PC Control

PC Control has four modes of operation: OFF, MONITOR, MANUAL and AUTOMATIC. Briefly stated, each mode only affects whether data acquisition and control inputs and outputs are active or not. The following table explains whether the computer **inputs** data from the acquisition devices or not and whether the computer **outputs** new values to a device or not.

Mode	Activities
OFF	–
MONITOR	Inputs
MANUAL	Inputs and Outputs
AUTOMATIC	Inputs and Outputs, with PID control

It should be noted that each mode does what the previous mode does and something more.

2.1 OFF Mode

When in the OFF mode, PC Control still allows you to use the menu structure and enter action commands, but its data acquisition and control systems are inactive (your screen will not show input tags being updated). This means that you control strategy can be programmed in when in OFF mode. In fact, *it is advised to make critical control changes, such as hardware assignments or programming Virtual (mathematics) tags, with the system in the OFF mode*.

2.2 MONITOR Mode

When you go from *OFF to MONITOR* an important step occurs: external devices are initialized. This is why, you cannot by-pass going to MONITOR mode first, before going to MANUAL or AUTOMATIC mode. In MONITOR mode, the following features begin working:

1. Inputs are read from the data acquisition devices
2. The input tag's values are updated
3. The alarm system begins working
4. The time queue begins working
5. The trend buffer is updated

Whether in OFF or MONITOR mode, whenever a state output or level output is changed, using the menu or action commands, the changes are made to the particular output tag. However, these outputs are not sent to the connected devices until the system mode is changed to MANUAL or AUTOMATIC mode.

2.3 MANUAL Mode

In MANUAL mode, changes made to state and level output tags (via action commands or the menu) are also sent to the appropriate output devices.

2.4 AUTOMATIC Mode

In AUTOMATIC mode, output tags are also sent to the appropriate output devices. But in addition, all level output tags that are programmed in a PID control loop mode, and individually in AUTOMATIC mode will have their output values calculated by PID equations. Notice that individual level output tags can be either in AUTOMATIC mode or in MANUAL mode. All level output tags in MANUAL mode will continue to be controllable manually via action commands or via the menu.

3 Obtaining System `Info` rmation

This section deals with PC Control's integrated system information reporting facility, obtained from the `System` → `Info` menu item. The purpose of the facility is to provide concise reports on all your active tags. The reports are available for view on the screen, or are printed out on a printer, so you can save it for your documentation. This facility complements the `Show` function of the tags editor, but instead of providing information about a single tag, all your active tags are referenced. As with the `Show` function, the `Info` function is very useful when configuring the tag parameters such as hardware, channels, gain and offset values, and so on. If you have several tag files, then you can make system information reports on each for your record keeping.

Each tag has too many details to show all at once, so the system information allows you to select one aspect of the tags (for example, hardware configuration) for its reports. In following sections, each of the categories of system information is described by providing examples of the screen output. As mentioned above, the `Info` menu item is available from the `System` menu item of the root menu. The next section shows how to select screen or printer for system information output.

3.1 Setting Screen or Printer `Destination`

After selecting the `Info` menu item, you will see a submenu (`Items, Destination`), from which, pick the `Destination` item. A prompt will ask you to select either the screen or printer for your info reports. Your choice is remembered by the system even after you leave the User Interface, by saving the option in the environment file, `PCC32.ENV`. Make sure your printer is ready if you have chosen the printer for output.

3.2 `Items` for System Information

The user must pick one of the tag information items, described in the following sections. As soon as you press <return>, the information is made available to the screen or printer.

Data Convert

Data conversion information for level input and level output type tags is provided. An example output follows:

```
Tag Name   Type        Gain      Offset     Limit    Mode        Rate/Decay
16:TI      L.I.    100.0000      0.0000              REPLACE
22:TI1     L.I.    100.0000     20.0000              REPLACE
32:PT3     L.I.     40.0000    -38.0000              REPLACE
33:FI      L.I.    500.0000     43.0000              REPLACE
34:LI      L.I.    100.0000    -18.2900              REPLACE
35:PDT     L.I.      4.0000    -20.0000              REPLACE
36:WTM     L.I.      0.0033      0.0000              ACCUMULATE
37:FO      L.O.      0.0999      0.0000        10
42:TO2     L.O.      1.0000      0.0000       100
46:TL      L.O.      0.0999      0.0000        10
50:TI1VIRT L.I.      1.0000      0.0000              FILTER           32.00
```

Set Points

The PID set point information is provided for level output tags. The information also shows setpoint ramping target and ramping period, if setpoint ramping is active. An example output follows:

```
Tag Name   Mode   Loop   Input Tag     Set Point      RPER        RTGT
37:FO      AUTO     1    33:FI           900.000
38:PO      AUTO     1    31:PI           291.000
39:LO      AUTO     1    34:LI            26.500
41:TO1     AUTO     1    22:TI1           70.000
42:TO2     AUTO     1    50:TI1VIRT       70.000
```

PID

The PID gain constants, and update rates are indicated for level output tags, along with the name of input tag in the PID control loop. An example output follows:

```
Tag Name   Loop   Input Tag    Prop. Gain   Int. Gain    Diff. Gain   Rate (Sec)
37:FO        1    33:FI        0.0100000    0.1000000    0.0250000         2
38:PO        1    31:PI        1.0000000    0.1667000    0.0417000         2
39:LO        1    34:LI        2.0000000    0.1250000    0.0313000         2
41:TO1       1    22:TI1       1.0000000    0.0250000    0.1000000         4
42:TO2       1    50:TI1VIRT   1.0000000    0.0250000    0.1000000         4
```

Alarms

Shown are alarm trigger points, alarm enable/disable status, and alarm action command. An example output follows:

```
Tag Name   Tag Type      Type      Value         Action         Enabled
11:SYSPWR  State Input   Lo               [                    ]    NO
15:S2.4    State Input   Lo               [                    ]    NO
16:T1      Level Input   LoLo       0.000 [                    ]    NO
                         Lo       315.000 [$T01 AU             ]
                         Hi       350.000 [$T01 MAN,$T01 DATA 0]
                         HiHi     400.000 [EM FURNOFF          ]
22:TI1     Level Input   LoLo       0.000 [                    ]    YES
                         Lo        25.000 [                    ]
                         Hi       380.000 [$T01 MAN,$T01 DATA 0]
                         HiHi     450.000 [EM SHUTDOWN         ]
```

Hardware

All hardware configuration information is provided in concise form. This includes device type and channel/bit numbers. If the device is a virtual (mathematics) type, then the math equation is also shown. An example output follows:

```
Tag Name   Type   Device       Ch#    Port   Bit
 0:A1      S.O.  uMAC4000                1     0
 1:A2      S.O.  uMAC4000                1     7
 7:PS      S.O.  uMAC4000                1     6
 8:VAC     S.O.  uMAC4000                3     5
 9:LINPWR  S.O.  uMAC4000                3     1
10:ALLAMP  S.O.  uMAC4000                3     3
11:SYSPWR  S.I.  uMAC4000                0     0
16:T1      L.I.  uMAC4000        1
17:T2      L.I.  uMAC4000        2
22:TI1     L.I.  uMAC4000        7
31:PI      L.I.  uMAC4000       15
40:PR      S.I.  uMAC4000                0     0
41:T01     L.O.  CTC100          0
46:TL      L.O.  uMAC4000        3
47:KICKER  S.O.  uMAC4000                3     0
50:TI1VIRT L.I.  Virtual     Virtual equation is
 ( 1 * TI1     ) + ( 0 ) + ( 0 )
```

4 Tags Editor

4.1 Access to Tags Editor

To use PC Control's integrated tag editor, start at the root menu, (`System, Editors, Tune, Data, Graph, Quit`), from which, pick `Editors` and then `Tags`. The menu line should now display the menu selections (`Edit, Load, Save, Create, Delete`). The five menu items are all *file* functions. In other words, you edit, load, save, create or delete a *tag file*, having file extension `.TAG`. Remember that a **tag file** exists on your *disk drive*, and contains all the information of a set of 100 tags. It is loaded by PC Control into *memory*, which for brevity, we call a **`tag structure`**. When you first run PC Control, after turning on your computer, you will have also loaded a tag file. The tag structure is a copy in computer memory of the loaded tag file. It is important to understand the distinction between a tag file on your disk and the tag structure in memory. With that in mind, the menu items allow you to do the following:

`Edit` Edit the tag structure in memory. Any changes must be saved to the tag file if they are to be used at a later date.

`Load` Load an alternate tag file from disk.

`Save` Save the current tag structure in memory to a tag file. This function is usually called after a change is made to tags.

`Create` Make a new tag file. The user provides a new tag name.

`Delete` Erase a tag file from disk.

4.2 Making a New Tag File

Your system will have been delivered with (at least) one preconfigured tag file. You may find that you want a new tag file, with tag specifications changed for different purposes. Usually, the differences are few, for example, you may have alternative PID constants. To make a new tag file, the procedure is as follows:

1. Load given tag file when starting PC Control. The tag structure in memory is a starting point to your objective.
2. Use the file `Edit` menu item, to make all your changes. You may also have to use the `Tune` menu item, if you want to change, for example, the PID constants (see further in this chapter, for `Tune` menu).
3. Create a new tag file. Use the file `Create` menu item, and type in the new name of tag file when requested.
4. To save the changes you have made to your tag structure, you use the file `Save` menu item. You will be shown the list of existing tag file names on your disk to save to, one of which is the new tag file you just created. Make sure you save the tag structure to the correct tag file

name. This is really an *overwrite* function; the current contents of the file get overwritten by the tag structure. You can choose to overwrite an old tag file (missing the previous step), if that if your intention is to make a change to the old file.

4.3 Editing tags

Once you pick the tag editor (from the root menu `Editors` then `Tags`), the next menu line is (`Change, New, Delete`). These are the functions you can perform on *individual* tags, being one of 100 tags of the tag structure. There are 100 available tags in the tag pool. Every tag is either a 'deleted' tag or is one of the four tag types, state input, state output, level input or level output. The functions described below, allow a user to modify the contents of the tags in the tag pool:

`Change` To change a tag, for example, the gain setting of a level input (LI) type tag. This menu item also provides access to `Show`, which gives a on-screen summary of a given tag. Summaries are also obtained via the `Info` function - see Section 3. These summaries are very useful when re-configuring your system, and they can be printed out on a printer for your records.

`New` This menu item allows you to add a tag (from the pool of deleted tags) to all the ones you already have. A maximum of 100 are allowed, numbered 0 through 99.

`Delete` This menu item allows you to delete a tag from those that you have. By the term deleted, we mean that it is inactive. You will be shown a menu of all your tags, from which you pick the doomed tag.

When you pick `Change`, the menu line will be filled with tag names. A colon ':' separates the tag number (ranging from 0 to 99) from the tag (mnemonic) name. Move the highlighted item to the tag you wish to change, and press <return>. You will now be presented with a menu that breaks down the chosen tag into its components: (`Show, Name, Description, Display, Alarms, H/W, Data Cnvrt`). It is wise to read and remember what each menu item does, as shown below, so that you can quickly return to the menu items, whenever a certain change needs to be made.

`Show` displays a summary of the current tag information on the screen. All other menu items below allow the user to view/modify specific components of the tag. After modifying any component, it is a good idea to return to the show function to make certain that the desired change has been made.

`Name` allows you to edit the tag name. The tag name can contain letters and numbers and the underscore character _. A tag name has seven characters or less, and the user must ensure that all tags have different (unique) tag names, because tag command actions use the tag names to identify them.

`Description` allows you to enter a short phrase, or description of the tag. This description helps the user remember the sometimes abbreviated tag names. The description appears in the XY/Bar real-time graphics, and in the alarm log, amongst other places.

`Display` Tag display information is used for showing the engineering units that the data is displayed in. It is a string of characters, and its use varies with the type of tag. If the tag is of type

State Input— it can be programmed to show on the schematic screen, one string of characters if the tag reading is High, and another for Low.

State Output— it can be programmed to show on the schematic screen, `ON` when output is High and `OFF` when output is Low or vice versa.

Level Input— it holds the units of measurement being displayed on the screen. The units are shown for example in XY/Bar graphing.

Level Output— it holds the units of output. The units are shown for example in XY/Bar graphing.

`Alarms` allows you to view/change alarm settings. For details, refer to Section 9

`H/W` allows you to enter hardware assignment information. Hardware definitions are related to the system wiring and are determined by physical layout. The submenus of this item are:

`Device` One of a given set of supported devices

`Channel / Bit` Specific channel and bit number within the device

`Mode` (in the case of LI tags), is the mode of operation, being one of REPLACE, ACCUMULATE, DIFFERENCE or FILTER.

Note that your tag file is probably configured for good as far as hardware assignments are concerned. However, changes can be made easily using the above menu items. One of the hardware 'devices' is a virtual (or math) device. To modify a virtual tag's expression, see Section 12. For modes of operation see Section 4.4 below.

`Data Cnvrt` Level input and level output values are converted between display units and hardware units (or vice versa) using gain and offset specified by the user in this function. Level output tags some other additional parameters. See appropriate sections below on LI and LO tags.

4.4 Special Level Input Tag Specifications

Level input tags obtain their readings from a digitizing unit, for example from an analog to digital converter (ADC), and express the digitized value in engineering units for the user, by multiplying it by a specified **gain** and adding a specified **offset**. The engineering units are entered in a location accessed by the `Display` menu item. The units are shown, for example, in the real time XY/Bar graphs. Finding the gain and offset is explained by the following example.

An input device, generates signal readings of 4-20mA, representing 0 to 2000 psi of pressure. The gain required to show pressure in psi, is $(2000-0)/(20-4) = 125$. The offset needed is $-4(125) = -500$.

LI tags can be operated in one of four modes, settable via the `H/W` then `Mode` menu item: (`Replace, Accumulate, Difference, Filter`). Each mode differs in how it uses an incoming reading (value after gain and offset is applied to the raw signal) to obtain an updated data value for the LI tag.

`Replace` The simplest mode is replace mode. It just updates the old LI tag value with the latest reading. An example of an LI tag in replace mode would be one measuring temperature with a thermocouple.

`Accumulate` In this mode, the latest value of the LI tag equals the current tag value plus the incoming reading. An example use of an LI tag in accumulate mode is a 'wet test meter', that reads the change in gas volume, but since the tag is in accumulate mode, the tag reads the total gas volume.

`Difference` Used for finding the rate of change of a measured parameter. The difference mode finds the change in readings over a specified interval. The interval is in minutes, and is entered when you pick **`Difference`**. An example of an LI tag in difference mode is a weigh scale measuring the mass of a product. The change in mass over time is equivalent to measuring the product rate, provided appropriate gain and offset values are used. See Section 12 for details of a virtual tag in difference mode.

`Filter` This mode is similar to replace, but provides a way of removing fast variations, like spikes and noise from the measured signal. It is an implementation of a digital low pass filter. For details on its use, see Appendix D.

An important aspect of updating an LI tag's data value is what action PC Control takes when an erraneous reading is obtained from the data acquisition hardware. *In all modes, the old (previous) reading is used in place of the erroneous reading*. Hence, in accumulate mode, the new tag value is a predicted total, and in the difference mode, the new tag value is a predicted change.

4.5 Special Level Output Tag Specifications

Level output tags are restricted to have data values expressed as a percent of maximum. The output percent value is converted to obtain hardware device units, which the hardware actually understands, by the use of a defined **gain** and **offset**. In addition, the software allows a **maximum** limit to be placed on the output value. LO tags can be in Manual or Automatic mode. In the later case, a PID loop calculation automatically adjusts the output value. The **rate** of update via the PID calculation must be specified in this case. All this is explained with the use of the following example.

An output device, say a heater, accepts a signal of 0 to +5 volts to control its power. We have a 12 bit digital to analog converter (DAC) in our system that produces a voltage between 0 to +5 volts if you send it values of 0..4095. Therefore a 0% output should send 0 to the DAC, and 100% output should send 4095.

Four parameters, the gain, offset, max and rate must be set via the `Data Cnvrt` menu item: In our example, the gain is 4095/100 = 40.95. The offset is 0.0. The maximum value that the DAC can accept is 4095, so this is entered in via the `Max` menu item. The rate value is the interval in seconds used to determine how often the to update a process output value using the control block

PID equations: the tag output value is recalculated every RATE seconds. The rate value is not of issue if your LO tag is in Manual mode (PID control disabled).

5 Tuning the System

5.1 The `Tune` Menu

The Tune menu item is accessed from the root menu. Certain tag parameters can only be changed via the Tune menu. In general, long term tag configurations, such as hardware device, are made via the tags editor, while often changing parameters, such as LO tag output values can be viewed and changed via the Tune menu. Although tuning a control loop, which is described in detail in the next section, is only concerned with LO type tags, the Tune menu conveniently allows you to view/change all types of tags.

When you choose the Tune menu item, you are first asked to pick the type of tag from (`S.I.`, `S.O.`, `L.I.`, `L.O.`). Each of these menu items presents different functions, and many have action commands that provide alternatives to using the menus, as shown:

`S.I.` Shows the current state input value – whether at High or Low.

`S.O.` Shows the current state output value, and prompts you if you want it toggled from High to Low or vice versa. Changing the value of a state output can also be done via an action command:

`$tagname ON` or `$tagname OFF`

`L.I.` Shows the current value of the level input tag, and prompts you if you want to adjust the value. This is done by adjusting the offset value so as to match the tag value with the requested one. This adjustment can also be done by the action command:

`$tagname EQUALS value`

`L.O.` The level output tags can hold PID control information, if configured for it. To program the various control parameters, you are shown a submenu with items:

`Data` Shows you the current output value (in percent). A prompt is shown, where a new value for the LO tag can be entered. This menu item has an equivalent action command: `$tagname DATA value`

`Mode` The operating mode of a LO tag is manual – in which case it's value can be changed via the `Data` menu item, or the mode is automatic – in which case the output is calculated by a PID control equation. This tag toggle is superseded by the system mode, which can be in off, monitor, manual or automatic modes.

`Loop1` This is where PID control parameters can be examined/changed. More on this follows in the following sections.

A host of action commands achieve the equivalent functions of the `tune` menu, as shown in the table below.

$*tag* AU(TOMATIC)	turn PID control on
$*tag* MA(NUAL)	set to manual (no PID)
$*tag* DATA *f*	change output to *f*
$*tag* KP *loop f*	see note below
$*tag* KI *loop f*	see note below
$*tag* KD *loop f*	see note below
$*tag* SET *loop f*	change set point to *f*
$*tag* SET+ *loop f*	adds *f* to current set point
$*tag* RPER *loop f*	make set point ramp period *f*
$*tag* RTGT *loop f*	make set point ramp target *f*
$*tag* OF(F)	turn SO tag off
$*tag* ON	turn SO tag on
$*tag* EQU(ALS) *f*	see note below

Notes:

$*tag* EQU(ALS) *f* — Sets level input tag to *f*. If in replace mode it calculates new offset. If in accumulate mode it sets the tag to this value. Not allowed in difference mode

$*tag* KP *loop f* — Sets the proportional gain to *f*. Use with caution since this affects the PID calculations

$*tag* KI *loop f* — Sets the integral gain to *f*. Use with caution since this affects the PID calculations

$*tag* KD *loop f* — Sets the differential gain to *f*. Use with caution since this affects the PID calculations

A full list of action comands is given in Appendix B.

5.2 PID Control and Tuning the System

Tuning refers to setting of control parameters kept within a level output tag which is in a PID control loop, maintaining control of a specified level input tag. Before PID control is set up, the user is urged to read section 17 of *Getting Started* chapter, which describes how PID control works. PC Control's PID algorithm ins detailed in Appendix C.

5.2.1 Setting up a Control Loop

A control loop is defined by assigning a level input tag and the PID coefficients to the control block of a level output tag. The control loop must also be enabled, and the LO tag placed in automatic mode. These parameters are accessed by way of the **Tune** menu item from the root menu. Note that only loop 1 of the three possible loops should be used. Loop 1 must have 100% weighting. No support for loops 2 and 3 is provided, and future versions of PC Control will remove multiple loop option.

Before automatic control of an output tag can be achieved, a number of conditions must be met:

- At least one control loop in the tag must be active with at least a non-zero proportional gain and preferably a non-zero integral gain. These control block parameters are accessed through the `Tune → L.O.→ tagname → Loop1` submenu.
- The LO tag of the control loop must be set to AUTOMATIC instead of MANUAL mode. This may be done from the `Tune → L.O.→ tagname → Mode` submenu, or by issuing an action command:

  `$tagname AUTOMATIC`
- The system operating mode itself must be set to AUTOMATIC through the `System → Mode` submenu or by issuing the action command

  `AUTOMATIC`.

Note that the system operating mode, when in AUTOMATIC, will cause all LO tags in control loops, that are individually also in AUTOMATIC mode, to use the PID control loop to determine its output.

5.2.2 Editing the PID Coefficients

The coefficients of a PID equation represent the proportional, differential and integral gains used in the calculation of the output value. The `Tune → L.O.→ tagname → Loop1` provides access to the control loop enable/disable function, the three PID coefficients for Proportional, Integral, and Differential gain, and the set point ramping parameters.

The rate variable, which can be examined/set from the

`Editors → Tags → Change — tagname → Data Cnvrt → Rate`

menu item controls the frequency of PID calculation and tag value update, i.e. the output is recalculated every rate seconds.

5.2.3 Tuning the Loop

Once a tag set has been created and edited to reflect the configuration of the system, and the control loops defined, the loops themselves must be "tuned". A tuned loop is one that can not only maintain the setpoint with minimal error, but can react as quickly as possible to disturbances with little or no overshoot.

The following method for control loop tuning is due to Ziegler-Nichols[2], which is based on frequency response analysis of a *closed loop* system, with P, PI, or PID control. To achieve just P or PI type control, the user simply zeroes the appropriate unused gain constants, K_I and K_D. The following steps are required to test the control loop characteristics:

1. Set up the control loop, as described in Section 5.2.1. In this step, the LO tag must be in manual (no PID) mode. Note the current set point value, which should be your desired LI

---

[2] "Optimum Settings for Automatic Controllers," by J. G. Ziegler and N. B. Nichols, *Trans. ASME, 64*, 759 (1942).

tag value. Bring the desired input point close to the current set point by manual control. This is most easily done with the `$tagname DATA n` action command for the LO tag, where the n output percent value is adjusted from time to time.

2. Use just proportional control, i.e. have differential gain at zero, and integral gain at zero, or a very small number like 0.0001 if you experience difficulties in tuning. The integral gain provides DC level recovery, as described in Section 17 of *Getting Started* Chapter. The proportional gain can start at 1, and increased or decreased later as described in the next step. Calculations of the PID algorithm occurs at programmable rates: Set the **update rate**, T in seconds, of the PID control loop as described in the previous section. Choose a small update rate (eg. 2.0 seconds) when you have a fast responding system, as in pressure control by a valve, and a larger rate (eg. 20 seconds) when you have a slower responding system, as in temperature control of a large object. Then turn this loop to automatic mode, enabling PID control.

3. Force the system into oscillation, i.e. get the LI tag value to oscillate, by imposing a load change. A load change is best obtained by making a small change to the set point. When you do this, ensure, if possible, that no other system changes are occurring, otherwise you may get false results. Note the change in **set point change**, C. The change can be either upwards or downward from the current set point (try both, in different experiments). This is easily achieved using the `$tagname SET+ 1 f` action command, where the f value is the change in the set point. Allow sufficient time after each set point change to observe the resulting effect. On fast reacting systems, changes are seen within a few seconds, as for example in an engine where you are controlling its engine speed. In slow reacting systems, changes might take hours, as for example in a chemical reactor, where you are controlling its temperature. The oscillations can most conveniently be studied with PC Control's graphics capabilities. If you don't obtain oscillation, then increase the proportional gain, K_P by reasonable amounts, until the system *just* begins to oscillate. Two quantities have to be computed, once you see the oscillations.

   From an observation of the oscillations, calculate the **amplitude**, A, which is half of the peak to peak difference in the LI tag value. The other parameter is the **period of oscillation**, T_{osc}. Measured in seconds, it is the period between two peaks of the LI tag value. After this step, you should have the values of:

Parameter	Description
A	Amplitude of oscillation in units same as the LI tag's
C	Change in set point in units same as the LI tag's
T_{osc}	Period of oscillation in seconds
T	Update rate of PID control loop, in seconds

4. Decide whether you want P, PI or PID control, and accordingly set the PID coefficients, K_P, K_I and K_D as shown in the table below:

Type of Control Desired	K_P	K_I	K_D
Proportional	$0.5C/A$	0	0
Proportional-Integral	$0.455C/A$	$1.2T/T_{osc}$	0
Proportional-Integral-Derivative	$0.6C/A$	$2T/T_{osc}$	$0.125T_{osc}/T$

5.2.4 Changing Set Points and Ramping Set Points

The setpoint for a control loop i.e. the target input value for the tag being controlled, is defined and used in the tag of the controlling output channel. It may be modified several ways: the first is through the menu item **`Set Point`** obtained via **Tune** → `L.O.`→ **tagname** → Loop1. The second way of changing the set point is by an action command:

```
$tagname SET 1 120.4 .
```

This action will change the setpoint in LOOP1 of a tag named 'tagname' to 120.4. To increment a set point from its current value, an action command, **`$tagname SET+ 1 f`** can be used, as mentioned in the tuning procedure above. The third way to change the setpoint of a loop is by way of setpoint ramping. With this facility, the set point can be made to change linearly from the current setpoint to a final **`target set point`** over a period **ramp period**. Both target set point and ramp period (specified in minutes) must be specified, and they are also accessed via the Loop1 menu item of the **Tune** menu. As time progresses, the set point is ramped to its target value, and the ramp period decreases. *A ramp period of 0 indicates that set point ramping is inactive.* So, the target set point should be programmed first, *and then* the ramp period. A set of action commands allow the user to set the ramp period and target set point:

```
$tagname RTGT 1 400.0
$tagname RPER 1 4.0
```

These action commands make the set point for the control loop 400.0, and the ramp period 4.0 minutes.

5.2.5 Viewing Loop Response

The response of process variables to changes in control loop parameters may be conveniently observed using the two graphics functions: XY/Bar real-time graphics and the Historical trending, see Section 11. Process variables may also be logged to a disk file, see Section 11.

6 Screen Editor

A **screen file**, `SOME.SCR` is loaded when you invoke the batch program, `GO.BAT` for PC Control with the parameter `SOME`. In `SOME.SCR` are contained the character id numbers, character-colour id numbers, and positon information of tag values. Integrated within PC Control is a screen editor that allows you to define your own schematic diagram. Since your screen is eventually stored on a disk file, it is easy to create several screens with different views of your process. At any given time, you can choose to load an alternate screen, by using the `Load` menu item of the screen file editor, thus offering multiple views of the process.

6.1 Using the Screen Editor

To use PC Control's integrated screen editor, start at the root menu, (`System, Editors, Tune, Data, Graph, Quit`), from which, pick `Editors` and then `Screen`. The menu line should now display the menu selections (`Edit, Load, Save, Create, Delete`). The five menu items are all *file* functions. You can edit, load, save, create or delete a *screen file*, having file extension `.SCR`. The five menu items do the following:

`Edit` Edit the screen file that is already loaded into PC Control.

`Load` Load a screen file. This replaces the previous screen with this new one.

`Save` Save a screen file. After you finish editing a screen file, remember to save your work!

`Create` Create a new screen file. This is the first step in making a new screen file.

`Delete` Erase a screen file from disk.

6.2 Creating, Loading and Saving a Screen file

If you are starting afresh, you simply create a new screen file, using the `Create` menu item. You will be prompted for a name of a screen file, and then the program will create a new screen file with as yet undefined display contents. Next you would load that file, clear the display, and edit the display. Finally you would save the screen file. Sometimes, you will be modifying an existing screen file, or making a second screen file from an existing one, with a few changes. The steps to follow in that case are obvious: You load a screen file, edit and then save to the appropriate screen file name. *Whenever you save a screen file, you are actually* **overwriting** *the contents of the chosen file*. After editing a screen file with the editor, it must be saved back to a screen file, so that you can use this screen file again after the computer is switched off.

6.3 Screen Editing Functions

When you pick the `Edit` menu item, the current display changes to one having a similar appearance: the schematic diagram is still visible, but where you usually see the tag values being updated, you now see the tags' numbers, which are shown as two digits 00 through 99 in reverse video, and flashing attributes. *The flashing numbers, are tag numbers, and are the designated locations where the tag values have been programmed to appear*. You will find a cursor flashing on the schematic. The current cursor location is where you can enter in the letters, numbers and graphics characters that make up the screen. If you type some message, like '`temperature`', it will be enterd onto the screen at the cursor location. The cursor can be moved using the cursor direction keys, <left>, <right>, <up> and <down> keys. *The screen editor memorises the direction key you last pressed, and will use that direction whenever you type in a message*. This is a convenience when typing in a series of graphics characters.

The function key usage has been redefined for screen editing functions, as explained below.

F1 help key

By pressing function key <F1> you are provided with on screen help for the screen editor.

F2 paste function

To print a graphics character, (a list of all characters, of ASCII numbers 0 through 255 can be found in the IBM technical reference manual), it must first be assigned to one key of the keyboard. Suppose it is the key a. Then instead of producing an '**a**' this key prints the graphics character. The assignment of a key to a graphics character is done via the function **paste**.

It is also through the paste function that you select a **colour** for printing to the screen. *Note that when you first invoke the Screen Editor, the keys are assigned their normal characters, and the colour for printing is white*.To place a message in colour, position the cursor where you want the message, then press the paste function key, <F2>. This allows us to paste a character with a given colour.

At the bottom right of the screen is a graphics character whose colour is white. Press <up> or <down> keys, while observing the colour of the character change. When you have seen one you like, press the \ key (this is a character you won't use often). A beep will sound. This tells us that we have 'pasted' the \ key with the graphic character. In addition all characters typed in now will your chosen colour. Press <esc>. Now if you type in '`weight`', you will observe that it is printed in your chosen colour. If you type \, then our graphic character will be printed, also in the chose colour. The screen editor memorises the last colour chosen.

While in the paste function, <F2>, if you press the <left> or <right> keys, you will notice that the character to paste changes. In fact the characters are ordered by their ASCII numbers, as listed in the IBM technical reference manual. You can pick the graphic symbol by pressing <left> or <right> keys and paste it to any key, and then print that character on the screen. In this way it is possible to create a custom process schematic with the extended ASCII character set.

F3 place tag function

Once the schematic has been drawn, the tag's values can be programmed to appear 'on top' of the schematic by using the <F3> key. *Each tag can be placed in only one screen location*. To do this, first position the cursor to where you want the left end of the tag data. Next press <F3>, and then you are presented with a menu of all the tags of your system. When you have picked one and pressed <return>, you have assigned that tag to be displayed at the cursor position. If you happen to be displaying a level input or level output type tag, you will also be shown two prompts,

```
Enter Field Width For Data Display  [ 3 ] :

Enter # of Decimal Places For Data Display  [ 0 ] :
```

one after another to which you have to enter format information. At the first prompt, enter in **field width**, or the total number of spaces required to show the tag data. This includes the sign, integer part, fractional part, and the decimal point. At the next prompt, enter in the number of decimal digits after the decimal point. To figure this out, you need to the maximum width your data value will be, at the precision you want to display it.

For example, if you have a tag which you want displayed as `123.56` then you need 6 for the field width. But if you want to display `-123.56` then you need 7 for the field width. In both examples, you need 2 for the decimal places, to show the 56. Note that tag values are **single precision** values, and are correct to only about 6 digits. In addition, most digitizing instruments only provide 6 or fewer digits of precision. This means that displaying an excessive number of decimals is pointless. *If a data value cannot fit within the field width specified, it is shown on the screen as a series of asterisks: '******'.*

F4 clear screen function

If you wish to clear the screen and begin over, then press <F4>. Be careful with this function! **Hint**: When developing a new screen, periodically save your edited work, so that you can recall it if you have messed up your current screen.

7 Action Commands and the Error Log

Action commands are typed commands to the PC Control system that allow changes that are normally made through the menu items. There are three usual ways of getting action commands to execute. First, they can be typed in after pressing <F2> function key, and the result is an *immediate* change in the process. We can refer to this as an immediate mode. Second, action commands are programmed into the alarm system, and are automatically executed whenever an input point goes into an alarm level. Third, action commands are executed via the time queue as prescribed dates and times.

The <F2> function key provides a short cut to achieving an action without having to go through the menu structure to the menu item you want. The entry of action commands into the alarm system, and the time queues is discussed in later chapters. Any action commands can be typed in after pressing the <F2> key, i.e. you can 'test' your action command this way before using it in the alarm actions or time queue. The full set of possible action commands is listed in Appendix B. Several action commands can be executed together by putting them within a **macro**. Refer to Section 8 for more details.

7.1 Rules for Entering Action Commands

- Multiple action commands can be typed in by separating each action command by a comma ','. For example, you can change the system mode from OFF to MANUAL using an action command

  ```
  MONITOR, MANUAL
  ```

- Commands can be entered usually by their first few letters. Any number of extra letters, upto the maximum, as shown within the brackets in the tables in Appendix B are optional. For example, typing in any of

  ```
  MONITOR
  MONIT
  MO
  ```

  accomplishes the same thing, which is to turn system to monitor mode.

- Whenever a space is required between words of an action command, one or more space are acceptable. For example, both of the following actions are equivalent:

  ```
  ALARMS ON
  ALARMS  ON
  ```

7.2 Syntax Checking, Error Messages and Error Log

Each time you enter an action command — either in the immediately executable mode after typing <F2>, or in alarm actions, or time queues, or macro actions, PC Control performs a validity check on the action command. Any inconsistencies are reported as errors. If the User Interface is active, then the errors show up at the bottom of the screen. Sometimes, several errors can occur one after another, and only the last error message is seen at the bottom line of the screen. Fortunately, all error messages, and their date and time of occurrence are recorded by the background `CONTROL` program. To view the error messages, you can press <F7> from the User Interface.

In addition, the errors are logged to a disk file, in ASCII format, with file names of the form `MMDDHHMM.ERR`. The `MMDDHHMM` stand for the month, day, hour and minute of the start of error log. The error log of one run may be split into several files, and all error files are kept in the `ERRORS` subdirectory of `PCC` directory. All possible error messages are explained in Appendix A. Note that error messages may also be issued whenever a hardware related problem occurs. These errors are *also* recorded in the error log.

PC Control provides an extremely useful feature for checking action commands for **syntax** errors. To check, for example, that your alarm action commands do not contain any spelling mistakes, or refer to wrong tags or non-existent macros, you can syntax check all tags having alarm actions. This is always a good idea before you take a long run with your system, only to find that there was a syntax error in one of your actions! Syntax checking is done via the following menu selections: `System` then `Check`. This menu then presents a choice of:

`Macros` Syntax check all action commands in all macros, in the macro buffer. The macro buffer has all the macros loaded in it and is described in Section 8.

`Time Q` Syntax check all actions currently in the time queue buffer. The time queue buffer has a previously loaded time queue in it and is described in Section 10.

`Alarms` Syntax check all alarm action commands present in the tag structure. The tag structure has a tag file loaded into it, and is described in Section 4.

`System` Syntax check all action commands. This is equivalent of doing all three cases above.

Once any of these items is picked and you press <return>, the background program of PC Control does a rapid syntax check of all the appropriate action commands. Depending on the performance of your computer, you might detect a brief pause in operation. It is advisable to do syntax checking with the system in the OFF mode, to prevent disturbing data acquisition. Any errors that the syntax checking finds will get reported to the *error log*. This log can be viewed by pressing <F7>. A sample error log after a syntax check is performed is shown below, and the contents of the log are explained following the log.

```
89/09/07 11:12:46                  ERROR  LOG

89-09-07 11:00:55 $ECMO DATA 120       : Percent out of range
89-09-07 11:01:00 *** Start of Macro Checking ***
89-09-07 11:01:00  $A1 ON              : Can't find tag: A1
89-09-07 11:01:00 $FO AU               : Can't find tag: FO
89-09-07 11:01:00 $PO AU               : Can't find tag: PO
89-09-07 11:01:00 $PS ON               : Can't find tag: PS
```

```
89-09-07 11:01:00 $LO AU              : Can't find tag: LO
89-09-07 11:01:00 Error in Macro: CONTINUE
89-09-07 11:01:00 *** End of Macro Checking ***
89-09-07 11:01:00 *** Start of Time Queue Checking ***
89-09-07 11:01:00 *** End of Time Queue Checking ***
89-09-07 11:01:00 *** Start of Alarm Action Checking ***
89-09-07 11:01:00 $ECM00 DATA 0       : Can't find tag: ECM00
89-09-07 11:01:00 Bad alarm action(s) in Tag: DIN2
89-09-07 11:01:00 *** End of Alarm Action Checking ***

                          Press Any Key To Continue
```

The first line shows the current date and time, and a title of the screen – ERROR LOG. Following lines show error messages that are date and time stamped at the time of occurrence of the errors. The first error message is `Percent out of range`, and is issued when a level output tag was being set to 120%. This error message occurred when the author typed in:

<F2> `$ECM0 DATA 120` <return>

Note that both the typed action command and the error message are shown. The rest of the error messages were a result of doing a syntax check on the system (all actions in the macros, time queue and alarms).

The error messages produced from syntax checking are enclosed by a `*** Start ...` and `*** End ...` caption. If, as in the case of the macro `CONTINUE`, errors are found, then the errors are reported first, followed by the name of the macro. Similarly, a bad alarm action command in a tag is shown by the error message (in above example, `Can't find tag ...`), followed by the tag in which that action command was used (in above example, `... in Tag: DIN2`). You can view about 40 lines of error messages, but remember that the full number of error messages are logged to an ASCII file in the `\PCC\ERRORS` directory. You can view these error messages conveniently with the public domain utility program `LIST.COM`.

8 Macros

Macros are a set of action commands to be performed one after another. Suppose you wish to carry out three actions: to shut a heater off, open a valve, close another valve. When such a series of actions need to be performed often, then the action commands need only be written once in the form of a macro. Each time you want to perform the series of actions, instead of typing in each action command, you simply replace them by typing in an action to execute the macro. Several corrective action commands for an alarm condition, for example, can be collected into a macro.

The advantages of using a macro include

- condensing a whole series of command actions into one, thereby reducing operator error during command entry
- reducing time needed to test frequently entered series of action commands
- a descriptive name can be given to a macro, eg FEED_OFF

All macros are loaded into a special area of memory, called the **macro buffer**, every time the batch file for PC Control is invoked. The macro buffer can hold upto 20,000 bytes of action commands, that will easily address all applications. The macro is given an 8 character name, being the name of the **macro file** in which it is saved. Macro files have extension `.MAC`. A macro can be executed by an `EMACRO SOMENAME` action command. The macro, `SOMENAME` is searched for in the macro buffer, and if found, it is executed. PC Control incorporates a macro editor, with automatic syntax checking of action commands, as your enter them.

The macro editor does not directly alter anything in the macro buffer, which is where PC Control finds the macro's action commands. Instead, the macro editor allows the user to modify or develop new macros that only reside on disk files. If macro files are changed (edited, created, deleted) via the macro editor, PC Control (User Interface) must be exited and re-run to load in the changed macros into the macro buffer. There is a special program just to load the macros — `L3MACROS.EXE`. A call to this program will be found in your batch file, `GO.BAT`.

8.1 Using the Macros Editor

To use PC Control's integrated macro editor, start at the root menu, (`System, Editors, Tune, Data, Graph, Quit`), from which, pick `Editors` and then `Macros`. The menu line should now display the menu selections (`Edit, Load, Save, Create, Delete`). The five menu items are all *file* functions. You can edit, load, save, create or delete a macro file, having file extension `.MAC`. It is important to realise that after editing macro files with the editor, they must be loaded into the macro buffer before they can be executed by an `EMACRO SOMENAME` action command. When the system executes a macro, it finds the associated action commands from the macro buffer, not the macro files. So it is important to know that the editor only edits macro files, allowing the following functions:

`Edit` Edit the macro file. It must first be loaded into the macro editor.

`Load` Load a macro file into the macro editor. Before you can edit a macro file, you must load one into the editor.

`Save` Save a macro file from the macro editor. This is the last step after you are finished editing a macro file.

`Create` Create a new macro file. This is the first step in making a new macro file.

`Delete` Erase a macro file from disk.

8.1.1 Creating, Loading and Saving a Macro file

If you are starting afresh, you simply create a new macro file, using the `Create` menu item. You will be prompted for a name of a macro file, and then the program will create a new macro file with blank entries. Next you load that file, and enter in your action commands. Finally you save the macro file, and then try it out by exiting from PC Control's User Interface, and then re-loading it with the batch file, `GO.BAT`. Sometimes, you will be modifying an existing macro file, or making a second macro file from an existing one, with a few changes. The steps to follow in that case are obvious: You load a macro file, edit and then save to the same file name, or to another file. *Whenever you save a macro file, you are actually* **overwriting** *the contents of the chosen file.*

8.1.2 Editing

```
 89/09/06 13:50:36                  Macro Editor                File: STANDBY.MAC

   0 [$A1 OFF                 ]
   1 [$F0 MAN                 ]
   2 [$P0 MAN                 ]
   3 [$P0 DATA 100            ]
   4 [$PS OFF                 ]
   5 [$L0 MAN                 ]
   6 [$L0 DATA 100            ]
   7 [                        ]
   8 [                        ]
            .
            .
            .
  18 [                        ]
  19 [                        ]

F1:Help   F2:Action   F3:Insert   F4:Delete
```

With the `Edit` menu item, you can edit a macro that has been loaded into the macro editor. To access this menu item, press <esc> key until you see the root menu. Then pick the following: `Editors` then `Macros` then `Load`. Pick a macro file to load. Now the contents of the file are loaded

into the editor. To examine the contents, pick the `Edit` item. The usual screen will be cleared, and you will see the macro editor screen which will look like the figure above.

Each action command is placed on a line. The command actions in this example turn heaters TO1 and TO2 to 100 percent heat output. The first action command will be highlighted. Any of the other action commands can be highlighted by using the up and down cursor keys. At the bottom of the editor, you will see the function key usage. For example, the highlighted action command can be entered or changed from its current entry by pressing <F2>, then typing in an action command, and then pressing <return>. As you enter them, syntax checking is automatically performed. If an invalid action command is used, an error message is issued (at the bottom of the screen). The function keys have the following usage:

F1 provides help information on the use of this editor.

F2 invokes a prompt from the user interface for the user to enter an action command. The entered action command is automatically syntax checked, thus allowing you to find and correct mistakes before you go on. Valid action commands are all of the ones shown in Appendix B. This includes executing another macro from within the current macro. This is referred to as **nesting**. Nesting can be as deep as 5 levels. By this we mean a macro calls another macro, that calls another macro, that calls another macro, that calls another macro, that calls another macro. Nesting levels greater than 5 cause an error, and will not produce the desired actions. Action command size is restricted to 19 characters in the macro editor. You can circumvent that by using abbreviations of action commands.

F3 is used to insert an action command in the middle of the list of action commands. After you press this key, a space is created at the highlighted position, and the remaining list of actions move down by one position.

F4 is used to delete an action command at the highlighted position. All remaining actions move up by one position.

9 Alarm System

9.1 Introduction

PC Control incorporates an alarm processing system which monitors both level input and state input tags. The tag values are compared against programmable **alarm setting** or **alarm trigger points** and if alarm conditions are met, an **alarm event** is caused. The alarm event is indicated visually on the screen by having the level or state input tag's value blink on and off. In addition, previously programmed alarm action commands for the tags are automatically executed. These actions can be programmed to be corrective measures, for example an action to change a related output tag to a new value. In an extreme situation, the alarm action command could be programmed to shut down your system entirely These two scenarios represent alarms used as a control mechanism, and as a safety feature. The level input and state input have the following alarm capabilities:

Level input Four alarms levels are offered, namely HiHi, Hi, Lo and LoLo. If the value of a level input tag becomes greater than the Hi or HiHi settings or becomes less than the Lo or LoLo settings, then an alarm occurs. The HiHi alarm limit is placed higher than or equal to the Hi limit. Similarly, the LoLo alarm limit is set at lower than or equal to the Lo alarm limit. This allows a further differentiation in the alarm limit placement, and each trigger point can have a different action command. If the HiHi and Hi alarm limits are not both required, then they can be placed at the same level. The alarm trigger settings must follow a certain logical order, as given below in Section 9.5.

State input One alarm setting is allowed – An alarm can be triggered if the state input reads a digital Low or a digital High. An alarm action command can be programmed for this setting.

Whenever an alarm event occurs, PC Control records the details in an alarm log. The alarm log is described below.

9.2 Enabling Alarms

The alarm processing for a given tag is done only when 1) the system is not in OFF mode, 2) global alarms toggle is enabled, and 3) the tag alarm toggle is enabled. No alarm processing is done when the system is in the OFF mode. It must be at least in MONITOR mode. The global alarms enable switch is like a master switch, and is accessed from the `System` → `Toggles` → `Alarms` menu item. An action command can also be used to turn global alarms on or off:

`ALARMS ON` or `OFF <return>`

The individual alarm toggles allow you to selectively turn alarms on and off for tags. These toggles are accessed from the tags editor: `System` → `Info` → `Items` → `Alarms` → `Enable`. Here again, an action command can be used to turn tag alarms on or off:

`$`*tagname* `ALARM ON` or `OFF <return>`

To view current alarm toggle settings, see next section.

9.3 Viewing and Setting Alarm Trigger Points and Alarm Actions

Before we can set any alarm trigger points, how do we know what the present alarm trigger points are? There are at least two ways of determining them:

1. We can use the `Info` feature of PC Control. This provides us with a summary of all tags. To access this feature, pick menu items: **`System`** → **`Info`** → **`Items`** → **`Alarms`**. The destination for summaries can be the screen or the printer. You can choose your output device from the `Info` and then `Destination` menu item. In both cases, you will see a summary that looks like:

```
Tag Name    Tag Type     Type      Value          Action             Enabled

11:SYSPWR  State Input  Hi                 [                    ]   YES
16:T1      Level Input  LoLo       0.000   [                    ]   NO
                        Lo       315.000   [$T01 AU             ]
                        Hi       350.000   [$T01 MAN,$T01 DATA 0]
                        HiHi     400.000   [EM FURNOFF          ]
```

2. The second way is using the `Show` menu item of the tags editor for looking at one specific tag. To access this, start at the root menu, then pick **`Editors`** → **`Tags`** → **`Edit`** → **`Change`** → *tagname* → `Show`. A drop down window presents some information about the tag, which includes the alarm setting, along with the corrective or emergency action command to perform should that alarm be set off. An example of the display is shown below:

```
          PI      : GAS OUT PRESS
          =======================

     Tag type = Level Input
     Alarm condition    =   NONE
     Device   = uMAC4000
     Channel = 15
     Data     =    464.844
     Mode     = REPLACE
     Conversion Gain    =   400.0000
     Conversion Offset =   111.0000
     Units               = psi.
     Alarms = ENABLED

   =========== ALARM  LIMITS ===========
   HiHi:  1500.000 : $A3 ON, EMAC FURNOFF
   Hi  :  1000.000 : $A1 OFF
   Lo  :    75.000 : $PS OF, $A1 OF
   LoLo:    50.000 : EM FURNOFF
```

To set an alarm trigger point, you can go about it either using the menu system, or by action commands. Using the menu requires picking **Editors** → **Tags** → **Edit** → **Change** → *tagname* → **Alarms** → **Limits**. If you are working with a level input type tag, you will get a further selection `Lo, Hi, LoLo, HiHi`. Pick one of the alarm types and enter in your alarm trigger point. Repeat this with other alarm types until all the alarm limits are done.

The equivalent action commands to set alarm trigger points, are indicated in the following table:

`$`*tag* `AL(ARM) ON`	turns alarming for tag on
`$`*tag* `AL(ARM) OF(F)`	turns alarming for tag off
`$`*tag* `LO` *f*	sets the Lo alarm limit to *f*
`$`*tag* `HI` *f*	sets the Hi alarm limit to *f*
`$`*tag* `LOLO` *f*	sets the LoLo alarm limit to *f*
`$`*tag* `HIHI` *f*	sets the HiHi alarm limit to *f*
`$`*tag* `ALLO`	sets SI tag to alarm if Low
`$`*tag* `ALHI`	sets SI tag to alarm if High

To set an alarm action commands, you follow a similar procedure to setting the trigger points. From the root menu, pick **Editors** → **Tags** → **Edit** → **Change**. Now, pick the tag you want and then pick items **Alarms** → **Actions**. If you are working with a level input type tag, you will get a further selection `Lo, Hi, LoLo, HiHi`. Pick one of the alarm types and enter in your alarm action commands. Repeat this with other alarm types until all the alarm limits are done. If you type in an invalid action command, then an error message will appear. This on-line syntax checking feature prevents you from entering bad action commands in the first place.

9.4 Syntax Checking Alarm Action Commands

To check that your alarm action commands do not contain any spelling mistakes, or refer to wrong tags or non-existent macros, you can syntax check all tags having alarm actions. This is always a good idea before you take a long run with your system, only to find that there was a syntax error in one of your actions! Syntax checking is done via the following menu selections: **System** → **Check** → **Alarms**. Once the **Alarms** item is picked and you press <return>, the backround program of PC Control does a rapid syntax check of all the alarm actions. It is advisable to do syntax checking with the system in the OFF mode, to prevent disturbing data acquisition. Any errors that the syntax checking finds will get reported to the *error log*. This log can be viewed by pressing <F7>.

9.5 Alarm Limits

The five possible alarm states for a level input tag are HiHi, Hi, Lo, LoLo, and no alarm. The action command parser will not let the HiHi alarm value to be set less than the Hi alarm value. Similarly the action command parser will not let the Lo alarm value to be set higher than the Hi alarm value or lower than the LoLo alarm. In other words we must have:

$\mathrm{LoLo} \leq \mathrm{Lo} \leq \mathrm{Hi} \leq \mathrm{HiHi}$.

Where ≤ means less than or equal to. If only one High alarm or one low alarm is desired then set the Hi alarm value equal to the HiHi alarm value or the Lo alarm value equal to the LoLo alarm value. This will trigger a HiHi or LoLo alarm only, so make sure the alarm action for HiHi or LoLo is set and not the alarm action for Lo or Hi. For those who have tag structures created with versions before version 3.2 then If the HiHi alarm value is less than the Hi alarm value or the LoLo alarm value is greater than the Lo alarm value only the Hi or Lo alarm will be active and the HiHi or LoLo alarms will never be triggered.

The three possible alarm states for a state input tag are Hi, Lo, and no alarm. State input tags can only alarm upon digital values of OFF or ON.

9.6 Alarm Log

PC Control maintains an alarm log, which is a record of the all the alarm events. This includes the date, time, and tag identity – tag name and description, and alarm type. When a tag first gets into an alarm state, that event is logged. When a tag stops being in alarm state, it again records that event. The alarm log is saved to an ASCII disk file, of format. The last 40 alarm events can also be viewed on screen by pressing <F8>. An example of an alarm log is shown below:

```
89/08/30 13:45:53                 ALARM  LOG

89-08-30 11:47:21  PI     :GAS OUT PRESS  Tag Value = 26497.14  Data = HiHi
89-08-30 11:47:29  TI2    :2ND OUT. TC    Tag Value =    18.71  Data = Lo
89-08-30 11:47:31  TI2    :2ND OUT. TC    Tag Value =    20.57  Data = OK
89-08-30 11:47:37  PI     :GAS OUT PRESS  Tag Value =   317.54  Data = OK
89-08-30 11:47:49  LI     :SIGHT GLASS    Tag Value =     9.42  Data = Lo
89-08-30 11:47:01  LI     :SIGHT GLASS    Tag Value =    -3.72  Data = LoLo
89-08-30 11:48:05  TI1    :TOP OUT. TC    Tag Value =    23.30  Data = Lo
89-08-30 11:48:13  TI1    :TOP OUT. TC    Tag Value =    26.47  Data = OK
89-08-30 11:48:25  LI     :SIGHT GLASS    Tag Value =    -1.20  Data = Lo
89-08-30 11:48:31  SYSPWR :SCREEN SIGNAL  Tag Value =     High  Data = Hi
89-08-30 11:48:33  LI     :SIGHT GLASS    Tag Value =    13.09  Data = OK
```

10 Time Queue Operation

10.1 Introduction

A **time queue** is a sequence of time/action command pairs describing a series of changes to be made by the controlling software at the prescribed times. The Queue is processed by the system as long as the operating mode is not "OFF" i.e. the mode must be set to MONITOR, MANUAL, or AUTOMATIC. There is a system toggle for time queue operation, accessed from the **`System`** → **`Toggles`** → **`Queue`** menu, which must be in the on position. The equivalent action command to toggle the time queue is: **`QUEUE ON`** or **`OFF`**. When the time queue is on, an indicator, **`QUEUE`** appears at the top of the screen.

A queue is created as a relative-time queue stored on disk in a text file having a **`.TMQ`** extension. Time queue files are loaded into a special are of computer memory called **time queue buffer**. When loaded into the time queue buffer, the times are no longer relative-times, but absolute times. How is this done? When a **`.TMQ`** file is loaded, the time delays saved to the file are converted into absolute times by adding them to the system time at the time of loading. These time conversions, are handled automatically by the built-in time queue editor, and the user does not have to concern himself or herself with relative-times because all times are displayed as absolute times. An example of a time queue as viewed by the **`Edit`** menu item of time queue editor is shown below:

```
 89/08/28 21:17:16            Time Queue Editor            File: HEAT.TMQ

   0> Aug  28 21:18        [ALARMS OFF               ]
   1> Aug  28 21:20        [$T01 DATA 50             ]
   2> Aug  28 21:21        [EMACRO HEATOFF           ]
   3> Aug  28 21:22        [                         ]
   4> Aug  28 21:23        [                         ]
   5> Aug  28 21:24        [                         ]
   6> Aug  28 21:25        [                         ]
   7> Aug  28 21:26        [                         ]
   8> Aug  28 21:27        [                         ]
   9> Aug  28 21:28        [                         ]
  10> Aug  28 21:29        [                         ]
  11> Aug  28 21:30        [                         ]
  12> Aug  28 21:31        [                         ]
  13> Aug  28 21:32        [                         ]
  14> Aug  28 21:33        [                         ]
  15> Aug  28 21:34        [                         ]
  16> Aug  28 21:35        [                         ]
  17> Aug  28 21:36        [                         ]
  18> Aug  28 21:37        [                         ]
  19> Aug  28 21:38        [                         ]

F1:Help   F2:Action   F3:Decrement   F4:Increment   F5:Retract-Q   F6:Advance-Q
```

Any action command from the P.C.Control repertoire may be used in a time queue, including the `QLOAD` *filename* action command, which causes the User Interface to load in *filename*`.TMQ` into the time queue buffer. This amounts to what is known as **chaining** of one time queue to another. Note: when specifying the time queue file to load by the `QLOAD` *filename*, you *only* specify the file name, and not the extension `.TMQ` as well.

10.2 How Time Queue Operates

Time queue processing is done by the background program, CONTROL, when the system is in MONITOR, MANUAL, or AUTOMATIC modes. Each time that the system time reaches the *top of every minute*, ie. when the seconds equal zero, the time queue processing begins. If the current time *matches or is in advance* of the time associated with the action command at the top of the time queue, that action command is removed from the top of queue and the action command is then executed. If an action command is removed from a time queue in this manner, then all the action commands below it move up in the queue by one position. If the current time matches or is in advance of several action commands, then all those action commands get removed and executed one after another in quick succession.

10.3 Time Queue Editor

To use PC Control's integrated time queue editor, start at the root menu, (`System, Editors, Tune, Data, Graph, Quit`), from which, pick `Editors` and then `Time Q`. The menu line should now display the menu selections (`Edit, Load, Save, Create, Delete`). The five menu items are all *file* functions. In other words, you edit, load, save, create or delete a *time queue file*, having file extension `.TMQ`. A time queue file that exists on your disk can be loaded by PC Control into memory, in an area called the time queue buffer. It is important to realise the difference between the time queue file (on disk) and time queue buffer (in memory) as we look at what each menu item does:

`Edit` Edit the time queue buffer in memory. A time queue must first have been loaded to edit it. Any changes can be saved to the time queue file if they are to be used at a later date.

`Load` Overwrite current time queue buffer with contents of a specified time queue file on disk.

`Save` Save the current time queue buffer in memory to a time queue file.

`Create` Make a new time queue file. User provides a new time queue file name. All its entries are blank.

`Delete` Erase a time queue file from disk.

10.4 Creating a Time Queue file

If you are starting afresh, you simply create a new time queue file using the `Create` menu item. You will be prompted for a name of a time queue file, and then the program will create a new time queue file with blank entries.

More often, you will be editing a previous time queue file, making a few changes, and saving that time queue. *It is important to save a time queue, before you execute it,* if you want to keep a record of it. Once the time queue is executing, action commands are being removed from the top of the queue, and you will have only a fraction of your original time queue if you choose to save it then! For this reason, it is advisable to use the time queue editor with the queue toggle off, otherwise you may get unpredictable results.

10.5 Using the Time Queue Editor

It is important to note that *the time queue editor allows you to edit the time queue buffer, not a time queue file.* However, after you are finished with editing the time queue buffer, you can save it to a time queue file. But before any editing can be done, you must load a time queue file. See previous section if you are starting with no time queue files. Otherwise, use the **Load** menu item to load one.

To enter the time queue editor, pick the `Edit` menu item. The normal screen will be gone, and you will be presented with a time queue editor screen. An example time queue, displayed by the editor, is shown above.

At the top left of the screen, is the system date and time. This is handy when you have forgotten what month it is. The menu bar is gone. At the bottom is a list of function key assignments. Studying the structure of the time queue entries the user should notice that each entry has an entry number followed by the month, day, hour and minute that the action command enclosed in square brackets is to be executed. Cursor arrow keys are used by *this* editor to move the cursor on the screen to the desired time queue entry (Up and Down) and to move the cursor to the associated month, day, hour or minute data (Left and Right). Other active keys are <PgUp> and <PgDn> which increment and decrement the entries displayed on the screen by twenty plus <F1> to <F6> which perform the editing operations. The function keys have the following use:

F1 is a facility to provide help information on the use of this editor.

F2 invokes a prompt from the user interface for the user to enter an action command. The entered action command is automatically syntax checked, thus allowing you to find and correct mistakes before you go on. Valid action commands are all of the ones shown in Appendix B. Action command size is restricted to 19 characters in the time queue editor, but you can circumvent that by 1) using an execute macro statement, that executes several actions, or 2) you can program several action commands in your time queue, all programmed for the same time.

F3 is used to decrement the parameter pointed to by the cursor. For example if the cursor was placed under the month, 'Aug', then pressing <F3> would change the month to 'Jul'. If the cursor was placed by the date, or hour or minute, you get a similar result for those items.

F4 is used to increment the parameter pointed to by the cursor, in a similar way.

F5 is the equivalent of <F3> as shown above except that all queue entries below the current queue entry (ie the entry with the cursor) are also decremented. Note that if the cursor is under the month all the following month entries are decremented but the day hour and minute entries are unaffected. Similar decrementing the day, hour or minute is possible by placing the cursor under the appropriate time component (ie day, hour, minute).

F6 is similar to the function of <F5> with the exception that the appropriate time component (month, day, hour, or minute) is incremented rather than decremented.

Note: *The time queue editor only allows you to program each event to be at the same or later time compared to the previous event above it.* This is the logic order of execution. **Tip:** The moment the time queue toggle is turned on, the action commands begin executing if their programmed times are up. So, a way of maintaining safe operation of time queues, while you are editing is to increment the date of each event by one. That way you have about 24 hours to make your changes! When the time queue configuration is suitable then the day can be decremented for all entries by placing the cursor under the day component of the first entry and pressing <F5>.

10.6 Chaining Time Queues

Chaining of one time queue to another allows one to have more than the 200 action commands limit allowed in one time queue. This is achieved by making the last command of one time queue a `QLOAD` *nextfilename* action command. A couple of important points must be noted when you want to chain time queues:

1. Since file loading is only done by the User Interface program, it must be active when the `QLOAD` action command is executed. This means that in a long run, or overnight run when a time queue is operating, and has a `QLOAD` action command in it, the User Interface must be left running, so that it can load the next queue file when the time comes.
2. The `QLOAD` command must be the *last* action command in a time queue to load another time queue. Suppose we have two time queue files, `FILE1.TMQ` and `FILE2.TMQ`, and we want to chain the first with the second. In the first time queue, we have a `QLOAD FILE2` command half way in the middle of it. If the `QLOAD` command is not the last action command in the first time queue, as in our example, then as the second queue is loaded, it will overwrite any remaining actions of the first queue. This means that after executing `QLOAD FILE2`, the second time queue file gets loaded, no matter where you were in the first time queue, and all the remaining actions in that old time queue are lost.

10.7 Repeating Time Queues

Sometimes, you may wish to repeat the actions of a time queue over again after one pass through the time queue. This is achieved by a similar technique to chaining a time queue as described above. Instead of loading another time queue at the bottom of one time queue, you write an action command to load the *same* time queue, i.e. place the name of the time queue you are executing in the `QLOAD` *queuename* command. By using this simple technique, you can cycle your system through several timed steps. Batch processing can be achieved using this scheme.

11 Data Graphing, Reporting and Logging

Introduction

PC Control provides the user with several data management facilities, all integrated into the User Interface. With these facilities, **tag values** can be graphed (to the display screen), reported (to disk file or printer) and logged (to disk file). More data management functions may exist in your version of PC Control (see Appendix section for operation of these features). Two graphing functions are the real-time **XY/Bar graphs** and **Historical trending**. The first provides the user with a continuously updating graphics display of up to 6 parameters (tag values) in separate windows. Each window can be toggled from an X versus Y plot to a bar plot at any time. The second plots 6 parameters on one large X versus Y graph. The reporting and logging features are discussed in following sections, but may help to remember that all of these facilities obtain data, the tag values, from one of two places (see Section 1):

1. Tag structure, where the current tag values are stored
2. Trend Buffer, where a set of 60 past values of each tag are stored

The trend buffer is central to both graphing and logging features, and it would be wise to read the next section to answer questions that you might have later.

How Trend Buffer Updates

In the trend buffer are recorded data values of only level input and level output type tags. State input and state output type tags cannot be trended (graphed). To understand the trend buffer, imagine it to be 60 rows of 100 tag values, as shown in the table below:

Sample #	tag 0	tag 1	tag 2	...	tag 98	tag 99
1	1013 p.s.i.	-12 V	20%	...	33.3 deg C	100%
2	1012	-12	21	...	33.7	100
3	1013	-12	21	...	33.5	100
⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	1012	-12	24	...	33.8	75
60	1011	-12	25	...	33.9	75

Each row contains the 100 tag values, and corresponds to a **sample**. The **trend sample period** is the time in seconds between each sample, ie. each row is separated by 'trend sample period' seconds. *It is the background program,* CONTROL *which automatically updates the trend buffer.* This means that the User Interface can be inactive for the trend buffer to continue updating. However, graphing and logging is only be done by the foreground program, User Interface. How does CONTROL maintain the trend buffer, a record of the past 60 values of all the tags? By copying the 100 current tag values into the trend buffer, replacing the oldest sample by the newest sample of data. It is a **cyclic buffer**, in that the buffer has no beginning or end. However, the program keeps track of which is the most recent sample (#60) and which is the oldest sample (#1).

The total time from the first to the sixtieth sample is known as the **Trending Interval**. The trending interval is a programmable parameter, as described below. If your version of software has an **acquisition sample period** of 2.0 seconds, then 60 acquisition samples take 2.0 minutes. This is the *smallest* value that the trending interval can be set to. The next higher value is 4.0 minutes. Each sample in the trend buffer is updated at *every other* acquisition sample obtained in the tag value. Note that the tag is still updated at every 2.0 seconds (the acquisition sample period), but the trend buffer copies the tag value once every 4.0 seconds (the trend sample period). The 60 samples are convenient -the number of seconds in a minute.

From the above description it will be clear that the trending interval can only have certain values: multiples of the sample period times sixty, in seconds. Or, in minutes, trending interval can be 2, 4, 6, and so on. The maximum allowed value for the trending interval is 1440 minutes (24 hours).

11.1 Setting Trending Interval

Move back to the root or main menu by repeatedly pressing <esc>. Select `Data` menu item. The next menu is (`Trending, Reporting, Logging`). Select `Trending`, and then `Interval`. The following prompt will appear:

```
Enter Time Base in minutes[ 2.00 ] :.
```

Enter a value that will be a multiple of the sample period, as explained above. If it is not a multiple, an error message will be issued. An equivalent action command to set the trending interval can also be use. To set the trend interval to N minutes, the (abbreviated) action command is

`FRE TREND` N.

Note: if the trending interval is changed, then the trend buffer is cleared. As each row of the trend buffer fills again, they will be separated in time by the new trend sample period.

11.2 Graphing Facilities

11.3 Setting up Graphing Parameters

You will already have used the graphing features, mentioned in the *Getting Started* chapter. By pressing <F3>, you get real time XY/Bar type graphs, and by pressing <F5>, you get historical trend plots on one large XY type graph. Although both these features are independent, they each rely on the trend buffer for data values to plot. To set the trend interval, refer to the previous section. In this section, we describe how to set parameters for plotting, such as axes limits. Both the XY/Bar and (historical) Trending features have similar menu selections, accessed via the root menu from the `Graph` menu item. Six channels are available for each, and an LI or LO tag can be chosen for plotting. From the `Graph` menu, you choose either `XY/Bar` or `Trending` parameters to examine/set, as described in the following sections.

11.3.1 XY/Bar

This is the real-time XY/Bar type graphing facility, that also provides a host of tag information in six independent graphics windows. After picking this menu item, you choose one of the six windows for which you want to change parameters. Next, you are presented with a menu of all the active level input and level output type tags. The currently chosen tag for this channel is already highlighted, and the user can decide to chose another instead. On pressing the <return> key, you have chosen that tag for plotting, and you can set the following parameters:

`Enable` This is a toggle that allows you to make this channel active or inactive. When inactive, only a blank space will appear in the selected graphics window.

`Scaling` The upper and lower plotting limits can be examined or changed. The limits are entered in the same units as the tag's data value. The program will not allow you to set the Upper Limit lower than or equal to the Lower Limit.

`Type` This toggle allows you to set XY type or Bar type graphing format.

The graphing parameters — trending interval, each channel's tag, plotting limits, and so on are recorded in the environment file, `PCC32.ENV`, each time you exit from the User Interface. Whenever you re-activate the User Interface, those parameters are read in from the environment file, so you can start where you left off. If the environment file is missing, then default values are used.

11.3.2 Historical Trending

After picking this menu item, you choose one of the six channels for which you want to change parameters. Next, you are presented with a menu of all the active level input and level output type tags. The currently chosen tag for this channel is already highlighted, and the user can decide to chose another instead. On pressing the <return> key, you have chosen that tag for plotting, and you can set the following parameters:

`Enable` This is a toggle that allows you to make this channel active or inactive. When inactive, this channel will have no trend line of the graph.

`Scaling` The upper and lower plotting limits can be examined or changed. The limits are entered in the same units as the tag's data value. The program will not allow you to set the Upper Limit lower than or equal to the Lower Limit. The Upper and Lower limits appear at the right of the graph, when you press <F5>.

11.4 Reporting

PC Control can provide the user with a **report**, which is can be described as a 'snapshot' of the system. All active tags of your system, which includes all four types of tags are reported in a concise format. An example of a report is shown below:

```
P.C.CONTROL DATA REPORT -  89/04/21 10:01:47
=============================================

STATE INPUT

 11:SYSPWR  On    // 12:S2.1     On    // 13:S2.2     Off  // 14:S2.3     Off  //
 15:S2.4    Off   // 40:PR       On    //

STATE OUTPUT

  0:A1      On    //  1:A2       Off   //  2:A4       Off  //  3:A5       Off  //
  4:A3      Off   //  5:S1       On    //  6:S2       On   //  7:PS       Off  //
  8:VAC     Off   //  9:LINPWR   On    // 10:ALLAMP   Off  // 47:KICKER   Off  //

LEVEL INPUT

 16:T1      :     72.300 // 17:T2      :     60.250 // 22:TI1     :     96.414 //
 23:TI2     :     75.963 // 27:T7      :     83.304 // 28:PT1     :    250.891 //
 29:PT2     :    259.904 // 30:PT4     :     22.705 // 31:PI      :    439.348 //
 32:PT3     : -50655.250 // 33:FI      :    502.230 // 34:LI      :     59.804 //
 35:PDT     :    -16.009 // 36:WTM     :   8190.692 //

LEVEL OUTPUT

 37:FO      :    100.000 // 38:PO      :      0.000 // 39:LO      :      0.000 //
 41:TO1     :     19.560 // 42:TO2     :     54.493 // 46:TL      :     20.000 //▪
```

The report begins with a heading, that also includes the date and time of creation of the report. Notice that there are four groups, each of a different tag type, with every tag's value separated by '//' marks.

The reporting function automatically generates reports at programmable intervals. The **reporting interval** is set in minutes, and can have a value between 2 minutes and 1440 minutes. The reporting function can be enabled or disabled, and when enabled, reports are created at reporting interval minutes.

Reports can be sent either to your printer, attached to the parallel port, or to disk files. When reporting to disk files, they are created and saved in the `\PCC\REPORTS\` subdirectory, with file name and extension: `MnDyHrMi.RPT`, where `MnDyHrMi` is the report creation date and time (as with the error log files).

Note: Reporting is only active in the foreground program, User Interface, and this program must be active when the next report time is up. If you exit from the User Interface while reporting is enabled, then you must re-activate the User Interface before the next report is due, otherwise, the following happens: A report is generated at once as you re-activate the User Interface, and then continues reporting at the original frequency — once every 'reporting interval' minutes, as if no interruption had occurred.

11.5 Setting up Reporting Parameters

11.5.1 Enabling Reporting

To enable or disable reporting, pick the `System` menu item from the root menu. Then pick menu items `Toggles` then `Reporting`. From this menu item you can toggle between reporting enabled or disabled states. The second way of toggling reporting is via the `Data` menu item as described below. An action command can also be used to enable and disable reporting: `REPORT ON` or `REPORT OFF`.

11.5.2 Setting Other Reporting Parameters

Move back to the root or main menu by repeatedly pressing <esc>. Select `Data` menu item. The next menu is `(Trending, Reporting, Logging)`. Select `Reporting`. Reporting parameters can be examined or changed via the following menu items:

`Status` This is a toggle that allows you to turn reporting on or off.

`Device` This is a toggle that allows you to direct reports to printer or to disk as files.

`Interval` After picking this menu item, you can examine or set the reporting interval.

11.6 Logging

PC Control provides the user with a **logging** feature, which produces ASCII disk files of tag data that can be conveniently loaded into Lotus©123 or Symphony, or compatible programs, using the 'number import format'. The log files contain data obtained from the trend buffer, that is upto 60 past values of tag data. The trend buffer has been explained above. Remember that the trending interval sets the interval between samples in this buffer.

The user must make a selection of tags from all active level input and level output types to log. This is a convenience, because you may not require logging of every tag. The selected tag numbers are specified in a special file in the default directory: `\PCC\VER32\PCC32.LTG`. This is the **Lotus Tag File**. You will require an editor to create such a file. An contents of an example `PCC32.LTG` file are shown below:

```
17 30 31 33 34 -1
<EOF>
```

This file causes the logging function to only log the data from the trend buffer, for tag numbers 17, 30, 31, 33 and 34. The `-1` must be placed after the final tag number to tell the program that is the end of the tag list. The `<EOF>` is the End Of File character, that the editor places at the end of a file. An example log file, as printed out to a printer is shown below:

```
"PC CONTROL LOTUS DATA REPORT" "" "" "" " 89/09/09 01:43:14" "" "" "Sampling

" Time " "T2"  "PT4"  "PI"  "FI"  "LI"
" (sec)" "2ND INNER T"  "FEED IN PRE"  "GAS OUT PRE"  "GAS FLOW IN"  "SIGHT G
    0.0,     4.6000,   22.2930,  339.0354,  445.1524,   29.1356
    2.0,     0.6250,   22.2635,  331.8702,  441.0754,   26.9450
    4.0,    -3.3500,   22.2341,  324.7050,  436.9984,   24.7544
    6.0,    -0.3657,   22.2046,  317.5398,  432.9214,   22.5638
                                   .
                                   .
                                   .
                                   .
  116.0,    41.5288,   20.5844,  182.5074,  208.6868,   78.7382
  118.0,    42.5590,   20.5549,  185.3058,  204.6098,   78.4619
```

The log file begins with a heading, that also includes the date and time of creation of the log. Only the specified tags in `PCC32.LTG` are logged. At the top of the sample values, is shown the tag name and description, which is obtained from the tag structure. These columns do not line up here, but once loaded into the spread sheet program, like Lotus 123, each column will be a different tag's value. If you are logging many tags, the log is split into blocks of upto 10 columns of tags. Each sample is separated in time by 2.0 seconds, because the chosen trending interval for the example was 2.0 minutes, with the first sample at 0 seconds, and the last at 118 seconds.

As in the reporting function, the logging function automatically generates logs at programmable intervals. The **logging interval** is set in minutes, and can have a value upto 1440 minutes, with the restriction that the logging interval be *greater than or equal* to the trending interval. If the logging interval is equal to the trending interval (the most common setting), then the following is true: The trend buffer fills with 60 samples of tag values, in 'trending interval' minutes, and these 60 samples are logged to disk. In all other cases, i.e. when the logging interval is greater than the trending interval, the program disregards some data in the trend buffer, and logs only the most recent 60 samples once every 'logging interval' minutes.

Logs are created and saved in the `\PCC\DATALOG` subdirectory, with file name and extension: `MnDyHrMi.PRN`, where `MnDyHrMi` is the log creation date and time.

Note: Logging is only active in the foreground program, User Interface, and this program must be active when the next log time is up. If you leave, or exit from the User Interface while logging is enabled, then you must re-activate the User Interface before the next log is due, otherwise, the following happens: A log is generated at once as you re-activate the User Interface, and then continues logging at the original frequency — once every 'logging interval' minutes, as if no interruption had occurred.

11.7 Setting up Logging Parameters

11.7.1 Enabling Logging

To enable or disable reporting, pick the `System` menu item from the root menu. Then pick menu items `Toggles` then `Logging`. From this menu item you can toggle between logging enabled or disabled states. The second way of toggling logging is via the `Data` menu item as described below. An action command provides a third way: To enable and disable logging use the action commands: `LOG ON` or `LOG OFF`.

11.7.2 Setting the Logging Interval

Move back to the root or main menu by repeatedly pressing <esc>. Select `Data` menu item. The next menu is `(Trending, Reporting, Logging)`. Select `Logging`. Logging parameters can be examined or changed via the following menu items:

`Status` This is a toggle that allows you to turn logging on or off.

`Interval` After picking this menu item, you can examine or set the logging interval.

12 Virtual Device Tags

12.1 Introduction

Virtual device tags are level input (LI) tags. However there is no physical measuring instrument such as a weigh scale associated with it. Instead, it is devised to have mathematics capabilities to manipulate the input values from real devices. *The virtual (math) device can be conveniently thought of as a calculator*. As an example, a virtual tag's value can be programmed to be the difference in temperature between the outside and inside walls of a heated cavity, as read by two level input tags. Or, it could be programmed to calculate the feed rate of material into or out of a system, by observing the mass change of the material in unit time. Both these examples are explained more fully below. Just as with any other level input tag, a virtual tag's value can be logged to files, or trended, or have alarm limits placed upon it. It can be the input tag for a control loop, thereby allowing tremendous flexibility in advanced control applications. The following is a step by step guide to programming PC Control for virtual tags.

12.2 Preparation

To begin with, one of the unused tags is made into a new tag, of level input type, with hardware assignment being virtual device. Go through the menus as below:

1. As always, tag editing should be done with the system in the OFF mode, to prevent inadvertent results. Type in the action command: <F2> `OFF` `<return>` to turn system to OFF.

2. From the root menu, pick `Editors` then `Tags` then `Edit` then `New`. Now pick the first available tag number, then choose level input (`L.I.`) from the four tag types.

3. Next supply a new tag name, and tag description. Press <esc> after this step to finish creating a new tag. Now that we have a new LI tag entry, we have to specify it to be a virtual device.

4. From the tags editor still, pick `Change` then `H/W` then `Device`. From this menu level, select `Virtual`. From this point, you proceed to enter the mathematical formula (see next section). So press <esc> for now, to do the next step first. You can later return to program in the equation.

5. From the `H/W` menu, now pick `Channel`. Here, you do not have to enter any; since it is not applicable for a virtual device. You may enter a `0` anyway.

6. Finally, from the `H/W` menu, pick `Mode` to select data acquisition mode. You may choose one of `(Replace, Accumulate, Difference, Filter)` modes. This will depend on what you want to achieve with the virtual tag, and is discussed below in the examples.

12.3 Entering the Mathematical Formula

Entering the equation for a virtual tag takes place after selecting **`Virtual`** as a hardware device. There are three **operands**, say a, b and c, and two **operators** to make up the equation. For example, in the equation

$$v = a + b * c$$

v is the virtual tag's data value, a, b and c are three operands that are the data values of other tags, and '+' and '*' are the two operators that may each be any of [+, −, * or /]. *Note that '*' and '/' have precedence over '+' and '−'.*

12.4 Operands

To increase the mathematical capability of virtual tags, each of the three operands has a programmable gain, or multiplying constant. Each of the three operands therefore must be entered as a separate multiplying constant C, and a tag number T. The constant C is a floating point number. Typically, one uses a constant value of 1.0, which means the operand equals the full value of the tag T. By making the constant zero, the operand value is zero, irrespective of the tag number, T.

Allowable tags are those of your set of tags – you are presented with a menu of tag names when selecting a tag. If a given tag is a level input or level output (LI or LO) tag, then the appropriate tag value is used. For all other cases, i.e. if the tag is a state input or state output (SI or SO), then the operand is assumed to be just the constant C. An operand combination can be any one of:

1. A constant C times an LI tag
2. A constant C times an LO tag
3. A constant C if tag T is neither an LI nor an LO tag.

Whenever an LI tag is referenced, it's current value is used. A virtual device tag is itself an LI tag, and may also be used; this amounts to what is called **chaining**. And whenever an LO tag is referenced, its current output value (in percent) is used. A virtual tag can be chained to do more complex math operations by making the present virtual tag use values from an earlier calculation of another virtual tag. ***At each sample period, all virtual tags are computed sequentially from low tag numbers to high.*** This means that the virtual tag with a lowest tag number gets updated first.

12.5 Operators and Overflow Protection

The three operands need two operators: `OP1` and `OP2` that are any of [+, −, * or /]. Care must be taken to avoid programming a division of a large number by a number close to zero; otherwise an overflow will occur. Division by any number less than 1.0E−10 is prevented by the PC Control program, giving a fixed error result: 9.999E+09. Overflows resulting from multiplying two large numbers can also occur, but is not monitored, and will cause the program to halt, so be careful in choosing the constants C. Note that '*' and '/' have precedence over '+' and '−', so that

$v = a + b * c$ is actually

$v = a + (b * c)$.

The next section illustrates by use of some realistic examples, how to enter in the mathematical equations.

12.6 Programming Examples

This section provides a by-example method of how to enter the parameters for virtual tags. The table below illustrates the eight parameters that make up the following expression:

```
V = (C1*T1) OP1 (C2*T2) OP2 (C3*T3)
```

where the C's are the multiplying constants (floating point numbers) in front of the tags T, and the OP's are the operators.

operand1			operand2			operand3	
C1	T1	OP1	C2	T2	OP2	C3	T3

It is a simple matter of mapping the math operators and operands of the expression you want to the appropriate virtual tag parameters, as we will see in the following examples.

Temperature Difference Between Inner and Outer Reactor Walls

Problem: The temperature difference measured by two thermocouples, TE401 and TE406 is desired. These LI tags will be picked from a menu of existing tags.

Solution: Make a virtual tag with the following constants C, tag numbers T, and math operators OP.

C1 = 1.0, T1 = TE401, OP1 = minus

C2 = 1.0, T2 = TE406, OP2 = plus

C3 = 0.0, T3 = KICKER

These parameters are summarized in the table below. We enter them in as follows

1. It is recommended to enter the math parameters when the system is not in the Monitor, Manual or Automatic modes. This way, the virtual device tags don't begin updating until the full math equation is programmed in. Changing math parameters on the fly, although possible, can cause unpredictable results, such as floating point overflow, which then might halt PC Control! So, change the system mode to OFF.
2. From the root menu, pick `Editors` then `Tags` then `Edit` then `Change` then `H/W` then `Device` then `Virtual`. The current mathematical formula is shown.
3. Press a key to see math formula editor.
4. Pick `C1` : type in `1.0`
5. Pick `T1` : Pick `TE401` from tag names shown

6. Pick `OP1` : type in a `-`
7. Pick `C2` : type in `1.0`
8. Pick `T2` : Pick `TE406` from tag names shown
9. Pick `OP2` : type in a `+`
10. Pick `C3` : type in `0.0`
11. Pick `T3` : Pick `KICKER`, an SO type tag from tag names shown. By choosing a SO type tag, we are stating that there is *no tag* in this position.
12. Press escape when completed. It is important to check what you have entered by going back to the `H/W` then `Virtual` menu.
13. We would like an updating value for this virtual tag, so select `Replace` from the menu `H/W` then `Mode`.
14. Just as with any other LI tag, we must also set the (overall) gain and offset, which will be `1.0` and `0.0` respectively. We do this by going through the menus: from root menu, pick `Editors` then `Tags` then `Edit` then `Change` then `Data Cnvrt`.
15. That completes the programming. To display this tag onto the screen, use the screen editor, and place the virtual tag in the location you want.

This example is summarized in the table below.

Update Mode = `Replace`							
Overall Gain = `1.0`							
Overall Offset = `0.0`							
C1	T1	OP1	C2	T2	OP2	C3	T3
`1.0`	`TE401`	`-`	`1.0`	`TE406`	`+`	`0.0`	`KICKER`

Heavy End Product Rate from a Hydrocracking Reactor

Problem: The product rate desired is the mass change in grams per hour at the heavy end outlet. A weigh scale measures the mass (in kilograms) of the heavy end sample, attributed to WT151, an LI tag, number 67. WT151 gets updated at every sample period, which is 2 seconds, but does not change significantly over short periods for an accurate measure of the rate.

Solution: Use a virtual tag, that is placed in the DIFFERENCE mode, which obtains its operand from WT151, and is programmed to find the difference of the mathematical expression every 30 minutes. An internal variable holds present and past expression values, which in our case will be the masses of the heavy end products at present and 30 minutes ago. Every 30 minutes, the virtual tag value will be updated by calculating the difference of those two masses. The answer is given in terms of kilograms per 30 minutes. To convert to grams per hour, we need to multiply our answer by 2000. This is achieved easily by making the LI gain to be 2000.0 for our virtual tag. These parameters are summarized in the table below. We enter them in as follows

1. Change the system mode to OFF, for reasons explained in the example above.
2. From the root menu, pick `Editors` then `Tags` then `Edit` then `Change` then `H/W` then `Device` then `Virtual`. The current mathematical formula is shown.
3. Press a key to see math formula editor.
4. Pick `C1` : type in `1.0`
5. Pick `T1` : Pick `WT151` from tag names shown
6. Pick `OP1` : type in a `+`
7. Pick `C2` : type in `0.0`
8. Pick `T2` : Pick `KICKER`, an SO type tag from tag names shown
9. Pick `OP2` : type in a `+`
10. Pick `C3` : type in `0.0`
11. Pick `T3` : Pick `KICKER`, an SO type tag from tag names shown
12. Next we place this virtual tag in the difference mode, by going through the menus: `H/W` then `Mode`: Pick `Difference`, then enter 30 (minutes) for interval.
13. Finally, we must also set the (overall) gain and offset, which will be `2000.0` and `0.0` respectively. We do this by going through the menus: from root menu, pick `Editors` then `Tags` then `Edit` then `Change` then `Data Cnvrt`.

This example is summarized in the table below.

<table>
<tr><td colspan="8">Update Mode = <code>Difference</code>, every 30 minutes</td></tr>
<tr><td colspan="8">Overall Gain = <code>2000.0</code></td></tr>
<tr><td colspan="8">Overall Offset = <code>0.0</code></td></tr>
<tr><td>C1</td><td>T1</td><td>OP1</td><td>C2</td><td>T2</td><td>OP2</td><td>C3</td><td>T3</td></tr>
<tr><td><code>1.0</code></td><td><code>WT151</code></td><td><code>+</code></td><td><code>0.0</code></td><td><code>KICKER</code></td><td><code>+</code></td><td><code>0.0</code></td><td><code>KICKER</code></td></tr>
</table>

Part IV

Appendices

A Error Messages

A.1 Hardware Related Error Messages

`ACL board error 0, ch:`*d* `st:`*x* — Stargate board produced an error while accessing channel *d* (in decimal), and status read from board is *x* (in hexadecimal). Record both numbers when reporting problem.

`ACL board error 1, st:`*x* — Stargate board produced an error while trying to set it into RUN mode. Record status read from board, shown as *x* (in hexadecimal) when reporting problem.

`DT2801 board error (hex):` *x* — An I/O error from Data Translation board 2801. Record the hex number, *x* (it is a 16-bit hi and low byte combined number), and refer to DT2801 manual for this error code.

`DT2815: Init timeout s=`*d* — Data Translation board 2815 failed to respond within time out period, when trying to initialize it. Record the status number *d*, and consult DT2815 manual for details.

`DT2815: Reset timeout s=`*d* — Data Translation board 2815 failed to respond within time out period. Record the status number *d*, and consult DT2815 manual for details.

`DT2815: Send lsb timeout s=`*d* — Data Translation board 2815 failed to accept the least significant byte within time out period, when trying to send level output value to it. Record the status number *d*, and consult DT2815 manual for details.

`DT2815: Send msb timeout s=`*d* — Data Translation board 2815 failed to accept the most significant byte within time out period, when trying to send level output value to it. Record the status number *d*, and consult DT2815 manual for details.

`Toledo: Bad data error` — This message appears when characters received from one of the Toledo weigh scales cannot be deciphered into a data value.

`Toledo: rx timeout error` — This message will appear when one of the Toledo weigh scales does not respond within a given time out period. Run diagnostic program if necessary.

`uMAC: Bad analog input` — This message appears when an incomplete level input data is recieved from uMAC4000. This is probably due to faulty communications. Run diagnostic program if necessary. (uMAC uses Stargate I/O board.)

`uMAC: Bad digital input` — A request for state input information from uMAC4000 returned bad data. Run diagnostic program if necessary. (uMAC uses Stargate I/O board.)

`uMAC: Can't send command` — The uMAC4000 device does not acknowledge after sending a command. Run diagnostic program if necessary. (uMAC uses Stargate I/O board.)

`uMAC: Not initializing` — The uMAC4000 device does not acknowledge an initialization command. Run diagnostic program if necessary. (uMAC uses Stargate I/O board.)

`uMAC: Not replying with data` — This error message appears if a communications problem with uMAC4000 exists. Run diagnostic program if necessary. (uMAC uses Stargate I/O board.)

`uMAC: tx not empty` — Stargate board is not transmitting characters to uMAC4000. Run diagnostic program if necessary. (uMAC uses Stargate I/O board.)

`uMAC digital input failure` — A request for state input data from uMAC failed. It probably means that the level input data will also not be received. (uMAC uses COM1)

`uMac NOT responding` — This message appears when uMAC4000 is not sending or receiving characters, or if a uMAC4000 device is missing. (uMAC uses COM1)

`uMAC response timeout` -- A timeout occured while waiting for an acknowledgement of a command sent to the uMAC4000 (uMAC uses COM1)

A.2 Software Related Error Messages

`*calculate_hardcoded_tag*` — An internal software error. Record this message and circumstances under which it occurs, and report to Atlantis Scientific.

`Can't find tag:` *badtag* — Action command made reference to a tag whose name cannot be found. A common error arising from typing errors!

`Can't execute command` — Action command cannot be executed.

`Can't set alarm Lo lower than LoLo` — An action command to set alarm trigger point must ensure LoLo limit is lower than or equal to Lo.

`Can't set alarm Lo higher than Hi` — An action command to set alarm trigger point must ensure Hi limit is higher than or equal to Lo.

`Can't set alarm Hi higher than HiHi` — An action command to set alarm trigger point must ensure HiHi limit is higher than or equal to Hi.

`Can't set alarm Hi lower than Lo` — An action command to set alarm trigger point must ensure Hi limit is lower than or equal to Lo.

`Can't set alarm LoLo higher than Lo` — An action command to set alarm trigger point must ensure Lo limit is higher than or equal to LoLo.

`Can't set alarm HiHi lower than Hi` — An action command to set alarm trigger point must ensure Hi limit is lower than or equal to HiHi.

`Can't set frequency <` *NN* `min.` — Action command to set interval is lower than the minimum of *NN* minutes.

`Can't set frequency > 1440 min.` — Action command to set interval is greater than 1440 minutes.

`CONTROL` *version info* — A greeting message, announcing version number when you go from monitor to manual mode.

`EQU not allowed in DIFFERENCE mode` — An action command EQU(ALS) to adjust an LI tag which is in difference mode is not allowed. The offset plays no part in a difference mode LI tag.

`Illegal command` — Just what it says – no such command or illegal command in present context.

`Incorrect number of parameters` — Action command has incorrect number of parameters. For example in `$tagname DATA 20 50`, 4 parameters instead of 3 were used.

`Invalid action for L.O. tag` — Action command for level output tag is not valid.

`Invalid action for L.I. tag` — Action command for level input tag is not valid.

`Invalid action for S.I. tag` — Action command for state input tag is not valid.

`Invalid action for S.O. tag` — Action command for state output tag is not valid.

`Log's interval can't be < trend's` — A command to set logging interval has too low an interval: it must be equal to or greater than the trend interval.

`Loop number out of range` — PID action command must use loop number 1, 2 or 3. (only 1 is supported).

`Must be a multiple of sample time` — A command to set an interval must be a multiple of the sample period.

`Nesting level too deep!` — This error tells you that an action command has too many indirect calls to parser. This can occur if a macro calls another macro which calls another macro ... (indirect calls over more than 5 levels are not allowed). Record circumstances, if problematic and report to Atlantis Scientific.

`No process loaded` — This error is produced if a tag file has not been loaded into PC Control, and a request to change system mode for MONITOR is made. System remains in OFF mode after error message. Load tag file, by rerunning PC Control with tag file name.

`Only legal change is to MONITOR` — Command tried to by-pass setting system to MONITOR first. A system must first be set to MONITOR mode, whose important function is also to initialize devices.

`Output exceeds limit:`*tagname* `=` *f* — This error occurs when a tag of type level is being assigned a value outside 0 to 100 percent.

`Parameter must be ON or OFF` — Action command must use only ON or OFF.

`Param must be RE or LO or TR or LT` — Action command expects a parameter, that can be any one of RE(PORTING), LO(GGING), TR(ENDING) or LT(REND) only.

`Parameter must be TM, MA, AL, or SY` — Action command for syntax check expects a parameter, that can be any one of TM(Q), MA(CROS), AL(ARMS) or SY(STEM) only.

`Parameter must be TR or LT or TM` — Action command for DUMP expects a parameter, that can be any one of TR(END), LT(REND) or TMQ only.

`Parameter` *N String* `is bad.` — Action command uses bad parameter. The parameter number and parameter that were deciphered are *N*, *String* respectively. For example in `$tagname SET OFF`, parameter 3, `OFF` is bad.

`*Parser error` *N*`*` — This error message, if it appears, will help pinpoint the location of an internal error which occurs while deciphering an action command. Record this message, noting the error number *N*, and circumstances under which it occurs, and report to Atlantis Scientific.

`Percent out of range` — Action command expects a percent value, ranging from 0 to 100 only.

`Present state requested` — Command tried to set system to mode it already is in.

`Tag command missing` — The action command is incomplete - tag name specified, but not the rest.

`Tag name must follow '$'` — Action command for tags did not use tag name after the $ character.

`Time parameter invalid` — Action command for WAIT has parameter that is out of range - negative or bigger than 18 jiffies. (18 jiffies is approximately 1 second).

`*update_li_tag error*` — An internal software error. Record this message and circumstances under which it occurs, and report to Atlantis Scientific.

B Action Command Syntax

The following is the full list of action commands allowed in PC Control. Note the two groups of action commands:

1. Those that operate on tags, actions begining with a **$** and
2. All other actions that operate on the system.

The notation in the following tables is as follows:

()	Letters within brackets are optional
mytimeq	Name of a time queue file (`mytimeq.TMQ`)
mymacro	Name of a macro file (`mymacro.MAC`)
f	A decimal number, like 54.5
n	Time in jiffies (18 jiffies is approximately 1 sec.)
tag	Name of tag
loop	PID loop number (use 1)

Commands can be entered ususally by their first few letters. Any number of extra letters, upto the maximum, as shown within the brackets in the tables are optional. For example, typing in

`MONITOR` or

`MONIT` or

`MO`

accomplishes the same thing, which is to turn system to monitor mode.

All actions requiring loop number, *loop* should only use loop 1 of the three possible loops. Loop 1 must have 100% weighting. No support for loops 2 and 3 is provided, and future versions of PC Control will remove multiple loop option.

B.1 System Commands

The following are system, or global commands.

`OF(F)`	turn system off
`MO(NITOR)`	turn system to monitor
`MA(NUAL)`	turn system to manual control
`AU(TOMATIC)`	turn system to automatic control
`AL(ARMS) OF(F)`	turn global alarms off
`AL(ARMS) ON`	turn global alarms on
`QU(EUE) OF(F)`	turn time queue off
`QU(EUE) ON`	turn time queue on
`QL(OAD)` *mytimeq*	load time queue file *mytimeq*
`EM(ACRO)` *mymacro*	execute the macro *mymacro*
`RE(PORTING) ON`	turn reporting to disk/printer on
`RE(PORTING) OF(F)`	turns reporting to disk/printer off
`LO(GGING) ON`	turns logging to disk on
`LO(GGING) OF(F)`	turns logging to disk off
`LO(GGING) ON`	turns logging to disk on
`FR(EQUENCY) RE(PORTING)` *f*	sets the reporting interval to *f* minutes
`FR(EQUENCY) LO(GGING)` *f*	sets the logging interval to *f* minutes
`FR(EQUENCY) TR(ENDING)` *f*	sets the trend interval to *f* minutes
`FR(EQUENCY) LT(REND)` *f*	see NRC-EDACOS version appendix
`WA(IT)` *n*	see section B.3 below
`CH(ECK) TM(Q)`	syntax check loaded time queue
`CH(ECK) MA(CROS)`	syntax check loaded macros
`CH(ECK) AL(ARMS)`	syntax check alarm responses
`CH(ECK) SY(STEM)`	syntax check all of the above
`DU(MP) TR(END)`	see NRC-EDACOS version appendix
`DU(MP) LT(REND)`	see NRC-EDACOS version appendix
`DU(MP) TM(Q)`	see NRC-EDACOS version appendix

B.2 Tag Commands

The following are tag commands. Note that tag commands are usually applicable to only one type of tag (LI, LO, SI or SI), and will cause syntax errors if used with a wrong tag type. A $ precedes the tag name, followed by the tag action command.

$*tag* AU(TOMATIC)	turn PID control on
$*tag* MA(NUAL)	set to manual (no PID)
$*tag* DATA *f*	change output to *f*
$*tag* KP *loop f*	see section B.3 below
$*tag* KI *loop f*	see section B.3 below
$*tag* KD *loop f*	see section B.3 below
$*tag* SET *loop f*	change set point to *f*
$*tag* SET+ *loop f*	adds *f* to current set point
$*tag* RPER *loop f*	make set point ramp period *f*
$*tag* RTGT *loop f*	make set point ramp target *f*
$*tag* OF(F)	turn SO tag off
$*tag* ON	turn SO tag on
$*tag* EQU(ALS) *f*	see section B.3 below
$*tag* AL(ARM) ON	turns alarming for tag on
$*tag* AL(ARM) OF(F)	turns alarming for tag off
$*tag* LO *f*	sets the Lo alarm limit to *f*
$*tag* HI *f*	sets the Hi alarm limit to *f*
$*tag* LOLO *f*	sets the LoLo alarm limit to *f*
$*tag* HIHI *f*	sets the HiHi alarm limit to *f*
$*tag* ALLO	sets SI tag to alarm if Low
$*tag* ALHI	sets SI tag to alarm if High

B.3 Notes on Special Commands

`WA(IT)` *n* — Causes the system to wait for *n* jiffies. There are 18.2 jiffies in 1 second. Use this command with caution, since it pauses the background control program

`CH(ECK) TM(Q)` — Syntax check the current time queue. All errors found are sent to the error log

`CH(ECK) MA(CROS)` — Syntax check the loaded macros. All errors found are sent to the error log

`CH(ECK) AL(ARMS)` — Syntax check all alarm actions. All errors found are sent to the error log

`$`*tag* `EQU(ALS)` *f* — Sets level input tag to *f*. If in replace mode it calculates new offset. If in accumulate mode it sets the tag to this value. Not allowed in difference mode

`$`*tag* `KP` *loop f* — Sets the proportional gain to *f*. Use with caution since this affects the PID calculations

`$`*tag* `KI` *loop f* — Sets the integral gain to *f*. Use with caution since this affects the PID calculations

`$`*tag* `KD` *loop f* — Sets the differential gain to *f*. Use with caution since this affects the PID calculations

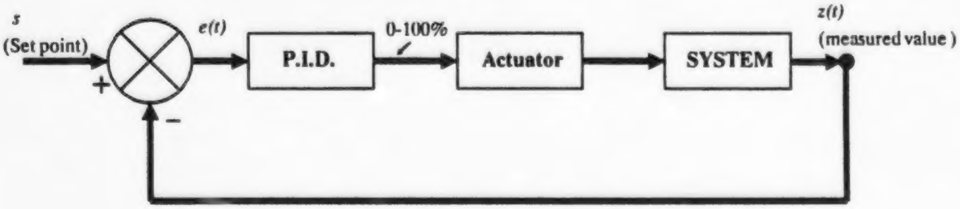


Figure 9: PID control loop

C PID Control Algorithm

C.1 Traditional Algorithm

PC Control employs proportional plus integral plus derivative (PID) form of automatic control. The control loop action can be described by the use of figure 9.

A system parameter is equivalent, in the real world, to a continuous measurement $z(t)$ of it. An example is a thermocouple that provides an indication of temperature of a vessel. The measured temperature variation is our input $z(t)$. The required value of the parameter is its set point, s. The summing junction at the left of the figure calculates the difference, or **error** signal, $e(t)$ between the set point and the measurement $z(t)$.

$$e(t) = s - z(t) \tag{1}$$

A PID control block calculates the actuator output $m(t)$, based on the error, $e(t)$:

$$m(t) = k\left[e(t) + \frac{1}{T_I}\int_0^t e(t) + T_D\frac{de(t)}{dt}\right] \tag{2}$$

where k is an arbitrary gain constant, the *proportional gain*. T_I is the *integral time constant*, or *reset time*, sometimes referred to as *minutes per repeat* (of the proportional action) and is often stated in minutes. T_D is the *derivative time constant* or rate time, also in minutes.

The three terms respectively are the proportional, integral and derivative components of the output. While this expression is true for a continuous input measurement, $z(t)$, and therefore $e(t)$, in practice, we only have discrete values of $z(t)$, at certain sample intervals. Hence, we use a discrete form of the PID equation:

$$m_n = k\left[e_n + \frac{T}{T_I}\sum_{j=0}^{n} e_j + T_D\frac{e_n - e_{n-1}}{T}\right] \tag{3}$$

where index n is the sample number, and T is the sample period between input measurements.

This is a more useable form of the equation, but note that the computation of m_n invloves a summation over n error values. Instead of computing m_n, we can calculate the change in the output, Δm

$$\Delta m = m_n - m_{n-1} \tag{4}$$

from which we can obtain m_n if we use a stored value of m_{n-1}. After substitution, we find

$$\Delta m = k\left[(e_n - e_{n-1}) + \frac{T}{T_I}e_n + \frac{T_D}{T}(e_n - 2e_{n-1} + e_{n-2})\right] \tag{5}$$

which is the equation employed in PC Control. Note that to calculate m_n, we need just four variables: e_n, e_{n-1}, e_{n-2} and m_{n-1}. In its implementation, PC Control assigns the 3 (programmable) constants for the PID equation as follows:

Proportional constant $K_P = k$

Integral constant $K_I = \frac{T}{T_I}$

Derivative constant $K_D = \frac{T_D}{T}$

The sample period T is also a programmable value in PC Control, called **update rate**[1].

C.2 Reset Windup Prevention

The contribution from the integral part (reset control) of the PID equation can sometimes grow out of bounds if an error signal exists, and stays uncorrected for an extended period of time. This leads to a very large, and unrealistic output value, m_n. To prevent this, the algorithm in PC Control clips the output m_n after PID calculation to maintain a 0 to 100 % range. This has been shown to effectively prevent reset windup.

---

[1] Specified in seconds, the update rate is really a period, not a rate!

D Low Pass Filter Mode for LI Tags

This mode of data collection can be chosen from one of replace, accumulate, difference, or the new filter mode, for level input (LI) tags. *The filter mode can be used to filter out rapid variation in the input signal and to remove spikes.* This filter mode implements a **digital low pass filter**. The exact form of this filter is the following:

$$y_i = Bx_i + Ay_{i-1} \tag{1}$$

with

$$A + B = 1 \tag{2}$$

Where A and B are constants, y_i is the new filtered tag value, y_{i-1} is the previous filtered tag value, and x_i is the current measured input value.

This filter has the effect of smoothing out high frequency changes. A programmable parameter, **decay time** defines the degree of smoothing. It represents the value $\frac{T}{B}$, where T is the sample interval of PC Control[1]. To program in the decay time, go through the menus as follows: `Editors` then `Tags` then `Edit` then `Change` then `(LI Tag)` then `H/W` then `Mode` then `Filter`. The user is then prompted with:

`Enter decay time (> 2.0 Secs).  Currently [ 8.00 ]:`

The current decay time is shown in the square brackets. Enter new value and press <return>. If the initial value of the input $x(0)$ is equal to 1 then our low-pass digital filter will be down to about 0.368 of an initial value after $\frac{1}{B}$ samples have passed, or after $\frac{T}{B}$ seconds have passed. This corresponds to the RC time constant of an RC low-pass analog filter. The "cutoff" frequency, f_c, by analogy is

$$f_c = \frac{B}{2\pi T} \tag{3}$$

Hence the larger the decay time the greater the smoothing.

---

[1] The sample interval is 2.0 seconds or 0.5 seconds, depending on your version of PC Control

Part V

The NRC-EDACOS Version

The NRC-EDACOS version of PC Control is an enhanced version of the normal PC Control. Described here are the enhancements.

E Installation Information

Please note the additional subdirectories for the NRC-EDACOS version must be created at installation time for PC Control to work. To install PC Control software, first follow the steps detailed in section 4 of *Installation*. Next, make the following additional sub-directories for your NRC-EDACOS version:

`C:\PCC\SUMMARY` — holds summary files - that are used to identify PC Control data with the High Speed Data Acquisition System.

`C:\PCC\DATA` — holds binary data files used by the Print-out facility.

`C:\PCC\TRANS` — holds Transient Test data files.

Details of these sub-directory's usage are given in the following sections.

F Panel/Remote Control and Bumpless Transition

This special feature is incorporated for the EDACOS version of PC Control. It provides, via software enhancements in the background program, a bumpless transition of throttle position and speed (or load) set point to Digalog controller unit whenever the user transfers from *panel to remote* or from *remote to panel* modes. By **panel mode**, it is meant control of devices are via potentiometers on the instrument panel. The **remote mode** refers to computer control of devices, via digital to analog converters. The purpose of having bumpless transitions is to prevent damage to hardware when control is transferred from computer to panel mode, each of which can be assumed to be at different output settings.

Panel Mode

PC Control can be either in PANEL or REMOTE mode. The state output tag 28:CNTRL determines which mode PC Control is in. On the digalog panel there is a LOAD/SPEED hardware switch, this switch has corresponding state input tags 18:LS_IN0 and 19:LS_IN1. These two tags have identical values and read the hardware LOAD/SPEED switch position.

When the state output tag 28:CNTRL is OFF we say that PC Control is in panel mode. When the LOAD/SPEED switch is in the **LOAD** position and we are in panel mode and PC Control is in AUTOMATIC or MANUAL mode, then the level input tag 3:THPOT is read in and sent out to the level output tag 52:ECM_O and the level input tag 4:SLPOT corresponds to the LOAD POT. on the digalog panel and controls engine load.

When the LOAD/SPEED switch is in the **SPEED** position and we are in panel mode and PC Control is in AUTOMATIC or MANUAL mode, then the level input tag 3:THPOT is again read in and sent out to the level output tag 52:ECM_O, but the level input tag 4:SLPOT now corresponds to the SPEED POT. on the digalog panel and controls engine speed.

Remote Mode

When the state output tag 28:CNTRL is ON we say that PC Control is in remote mode. When the LOAD/SPEED switch is in the **LOAD** position and we are in remote mode and PC Control is in AUTOMATIC or MANUAL mode, then the level input tag 3:THPOT is not sent out to the level output tag 52:ECM_O. Instead level output tag 52:ECM_O controls engine speed through PID control or setting the tag's value. The tag 4:SLPOT also no longer controls engine load, instead the level output tag 16:LD_SP controls engine load and the level output tag 17:SP_SP is left disabled.

When the LOAD/SPEED switch is in the **SPEED** position and we are in remote mode and PC Control is in AUTOMATIC or MANUAL mode, then the level input tag 3:THPOT is again not sent out to the level output tag 52:ECM_O. Instead level output tag 52:ECM_O controls engine load through PID control or setting the tag's value. The tag 4:SLPOT also no longer controls engine speed, instead the level output tag 16:SP_SP controls engine speed and the level output tag 17:LD_SP is left disabled.

Transition from Panel to Remote

When switching from panel to remote mode the tags 3:THPOT and 52:ECM_O may not be in agreement, also tags 4:SLPOT and 16:SP_SP or 17:LD_SP may also not be in agreement. To prevent engine speed and torque from changing too quickly when switching from panel to remote mode, PC Control sets the data value of the level output tag 52:ECM_O to the data value of the level input tag 3:THPOT and sets the data value of the level output tags 16:SP_SP and 17:LD_SP to the data value of the level input tag 4:SLPOT.

When the LOAD/SPEED switch is in the LOAD position the tag 52:ECM_O is set to control tag 0:ENGSP and tag 17:LD_SP is set to control the tag 1:ATORQ and tag 16:SP_SP is disabled. When the LOAD/SPEED switch is in the SPEED position the tag 52:ECM_O is set to control tag 1:ATORQ and tag 16:SP_SP is set to control the tag 0:ENGSP and the tag 17:LD_SP is disabled. PC Control also switches the tags 16:SP_SP and 17:LD_SP to manual mode and sets the tag 52:ECM_O to automatic mode. It is mandatory that the tags 3:SLPOT, 4:THPOT, 16:SP_SP, 17:LD_SP and 52:ECM_O are calibrated as percentages between 0% and 100%. If the level inputs tags are not in one-to-one correspondence with the level output tags, then the transition from panel to remote will not be bumpless.

Transition from Remote to Panel

As in the transition from panel to remote mode, the tags that control engine speed and load may not be in agreement with the potentiometer settings on the digalog panel. To prevent a jump in the engine speed and torque the level output tags 52:ECM_O, 16:SP_SP, and 17:LD_SP are ramped down to the level input tags 3:THPOT and 4:SLPOT data values and then PC Control is switched to panel mode.

PC Control takes one minute to ramp down linearly to the potentiometer settings. After one minute (120 sample periods) the tags 52:ECM_O, 16:SP_SP, and 17:LD_SP are set in manual mode and tag 28:CNTRL is switch to panel mode. *The LOAD/SPEED button should not be pressed while the tags are ramping down nor should PC Control changed to another mode while ramping down.* The level input tag 61:INDEX will count down from 120 to 0 while PC Control in transition from remote to panel mode and this tag can be used to determine when the ramping will be finished.

G Trend Log, Long Trend Log and Time Queue Log

G.1 Special Action Commands

The following action commands only apply for the NRC-EDACOS application.

`FR(EQUENCY) LT(REND)` *f* — sets the long trend interval to *f* minutes

`DU(MP) TR(END)` — dumps the trend buffer to the file `TRENDLOG.DAT` in directory `\PCC\DATA`

`DU(MP) LT(REND)` — dumps the long trend buffer to the file `LONGLOG.DAT` in directory `\PCC\DATA`

`DU(MP) TMQ` — appends the current data values of all the tags to the time queue log file `MMDDHHMM.DAT` in directory `\PCC\DATA`. Where the filename `MMDDHHMM` is the current month, day, hour, and minute.

The function key F9 issues a `DUMP TMQ` action command, resulting in the current tag data being appended to the current time queue log. The long trend interval can be set via the action command above, or equivalently it can be set or viewed via the menu: `Data` then `LongTrending` then `Interval`.

G.2 Trend Log

The trend buffer is a circular queue containing 60 entries. Each entry represent all 100 tag data values at the time of sampling. The **trend interval** is in minutes and represents the period of time over which the 60 entries are collected. The trend interval may be set with the `FREQ TR` *f* action command or via the menu: `Data` then `Trending` then `Interval`. The trend buffer can be dumped to the file \PCC\DATA\TRENDLONG.DAT using the `DUMP TREND` action command.

Being an action command, it offers great versatility. For example, `DUMP TREND` can be beneficially used as an alarm response, or in a macro with other action commands. This log file may be printed out using the printout facility to be explained.

G.3 Long Trend Log

In addition to the trend buffer a second data buffer called the **long trend buffer** *has been provided for EDACOS*. The long trend buffer is similar to the trend buffer but is not used for graphing purposes. The long trend interval contains 60 entries in a circular queue. Each queue entry represents all 100 tag values at the time of sampling. The long trend interval is in minutes and represents the period of time for 60 samples. Hence if the long trend interval was set to 60 minutes than each entry in the long buffer would represent a minute before the last entry. This buffer may be dump to the file \PCC\DATA\LONGLOG.DAT using the `DUMP LTREND` action command. This file may then be printed out at a later time using the printout facility.

G.4 Time Queue Log

When the time queue is turned on via the `QUEUE ON` action command or the **System** then **Toggles** then **Queue** menu entry, the file `MMDDHHMM.DAT` is created. Where `MMDDHHMM` is the current month, day, hour, and minute. Every time a time queue action is executed the time queue log is appended to with the data of all 100 tags before the command is executed. The log may also be appended to using the F9 function key or the `DUMP TMQ` action command. This file may then be printed out using the printout facility.

H Edacos Printout Facility

The printout facility is integrated into the User Interface of PC Control. *An attached LaserJet compatible printer is expected*, because an initialisation command string is sent to the printer (to set it to compressed mode).

H.1 Printout Format File

Format files contain all the content information used to produce a reports from. The files from which to make printouts are binary (and therefore unreadable) format. These files can be one of `TRENDLOG.DAT`, `LONGLOG.DAT` or `MMDDHHMM.DAT` files. They contain header and column information. The column information specifies column format and the tag to be print in that column. The header information contains the titles to be printed. The format files are all in the current directory and have the `.FMT` extension.

H.2 How to Obtain Data Printouts

From the **`Editors`** then **`PrintOuts`** menu selection you get the following menu, (**`Edit, Load, Save, Create, Delete`**). These entries all work on format files and work in a similar fashion to the other editors in PC Control. Once a format file has been Created and Loaded it may then be edited. From the **`Editors`** then **`PrintOuts`** then **`Edit`** menu selection you get the following menu, (**`Items, Samples, Header, New, Print`**).

From the **`Editors`** then **`PrintOuts`** then **`Item`** menu selection you get the following menu, (`Col1, Col2, Col3, ..., Col14`). From here you choose the column you want to edit. From the column menu you get the following sub menu (**`Set, Format, Clear`**) . The **`Set`** menu entry allows you to set the tag to be printed in that column. The **`Format`** menu entry allows you to set the field width and precision. The **`Clear`** menu entry allows you to clear a column.

From the **`Editors`** then **`PrintOuts`** then **`Samples`** menu selection you are prompted for the number of rows to be printed. The long log and the trend log have at most 60 rows, since these buffers hold at most 60 samples. The time queue log varies in sizes, depending on the number of time queue commands executed.

From the **`Editors`** then **`PrintOuts`** then **`Header`** menu selection you are prompted for the titles to be printed. There are three **`titles`** labelled title1, title2, and title3. They appear on lines one, two, and three of the report.

From the **`Editors`** then **`PrintOuts`** then **`New menu`** selection you can blank out all 14 columns of the current format file.

From the **`Editors`** then **`PrintOuts`** then **`Print`** menu selection you can print out any `*.DAT` file. You are given a menu of all the DAT files in the `\PCC\DATA` directory, just select the file you want printed and make sure the printer is ready. A report of upto 14 columns of tag data will be printed along with the file names of the format file and data file used to generate the report.

I Transient Test Facility

The transient test facility provides for a way of sending a sequence of pre-programmed values of engine speeds and loads to the engine. The load and speed values are sent every one second apart. There can be up to 1200 speed and load values in one sequence allowing for up to 20 minutes in a sequence. The sequence can be repeated by setting the **transient test repeat counter**. The test sequence is loaded from files in the directory `\PCC\TRANS`. The loadable files in `\PCC\TRANS` all have the file extension `*.OUT` and are created using the stand alone program **CONTRANS**, also supplied with the NRC-EDACOS version.

I.1 Creating Transient Files

In order to run a transient test sequence you must first load a transient file that has been previously created using the CONTRANS program. This program takes a text file `*.INP` which the user has created and converts it to a `*.OUT` file in directory `\PCC\TRANS`. The file to be converted must be a DOS text file created with any editor such as Borland's Side Kick©program, and the file should have the extension `*.INP` and must be in the directory `\PCC\TRANS`.

The `*.INP` file can contain up 1200 rows of speed and load data. Each row is made up of three fields. The first field contains the row sequence number. The row sequence number must be an integer from 0 to 1199 and should always *increase* from one row to the next. The sequence numbers allow for a row to be repeated without the user having to keep typing the same row over and over again. The second field contains the engine speed and the third field contains the engine load. The engine speed and load must be valid (Turbo Pascal©) floating point type numbers. The fields must be separated with spaces and each row must end with a carriage return. The following is a valid input file, `TEST1.INP`:

```
     0  2.5    5.3E+1   <CR>
     1  4.23   50.0     <CR>
     4  2.3    62.5     <CR>
     5  53.2   70.0     <CR>
<EOF>
```

The symbol `<CR>` represents the carriage return and `<EOF>` represents an end of file. In the above case notice that the sequence numbers for row two and three are 1 and 4 respectfully. The program CONTRANS will add sequence number 2 and 3 with the speed and load values of row two. Hence the contents of the above TEST1.INP file is equivalent to:

```
     0  2.5    5.3E+1   <CR>
     1  4.23   50.0     <CR>
     2  4.23   50.0     <CR>
     3  4.23   50.0     <CR>
     4  2.3    62.5     <CR>
     5  53.2   70.0     <CR>
<EOF>
```

Once a `.INP` file has been created it must be converted to a `.OUT` file so that it can be read by PC Control. This is accomplished using the CONTRANS program. Suppose the above file has file name `TEST.INP` and is in the `\PCC\TRANS` directory, then this file can be converted to `TEST.OUT` by typing

```
CONTRANS TEST.INP <return>
```

at the dos prompt. The extension `*.INP` will be assumed if it is left out. The file `TEST.OUT` can now be loaded with PC Control.

I.2 Loading and Running Transient Test Sequence

Once a `.OUT` transient file has been created with the stand alone program CONTRANS it can then be loaded and executed with PC Control. To load the file choose menu option **Editors** then **Transient** then **Load** from the main menu option. A file list of all `*.OUT` files in the directory `\PCC\TRANS` will be displayed. Select the desired test sequence and it then will be loaded into memory.

Make sure that the tag `28:CNTRL` *is in the REMOTE position and that the tags* `52:ECM_0`, `16:SP_SP` *and* `17:LD_SP` *are in manual mode.*

To run this test sequence select menu option **Editors** then **Transient** then **Edit** from the root menu. The prompt

```
Enter Transient Test Repeat:
```

should now be displayed. Select a number greater than zero. A zero for the transient test repeat variable indicates that the transient test sequence is not running.

A number greater than zero for the transient test repeat variable indicates the transient test facility is running and tells how many times it will repeat the test sequence.

Two special level input tags, `60:REPEAT` and `61:INDEX`, have been created to give information on the status of the transient test sequence. These tags should be displayed on the screen.

The REPEAT tag value reflects the transient repeat counter and counts down after each time through the sequence. The INDEX tag indicates where in the transient sequence PC Control is at. The INDEX tag is also used when switching from REMOTE to PANEL to tell how many half seconds of ramping down is left before the system is switch to panel mode. These two tags have no real devices and show up as device type 'Other'.

I.3 How to Stop Transient Test Function

A transient test sequence can be stopped at any time by entering zero for the transient test repeat counter from the **Editors** then **Transient** then **Edit** menu.

J Multi-Level Alarming

Introduction

The NRC-EDACOS version of PC Control incorporates a very versatile, multi-alarm processing facility in addition to the one that already exists.

The regular method of alarming on level input (LI) tags involve one tag triggering a HiHi, Hi, Lo, or LoLo alarm. *A multi-level alarm on level input tags involves up to three level input tags triggering an alarm.* This is achieved by programming a special level input tag, known as **VirtAlarm**.

A multi-level alarm is a type of virtual tag, since there is no physical measuring instrument such as a pressure gauge associated with it. Instead, it is a boolean expression of up to three other level input tags, which *do* have a physical measuring device. For example, a VirtAlarm can be programmed to trigger if thermocouple 1 and thermocouple 2 and thermocouple 3 are all over some programmed fixed temperatures.

The above example looks for one parameter being *greater* than another. In fact, PC Control's Multi-Level Alarm system allows five different conditional operators, as we shall see. The following is a step by step guide to programming PCC for virtual alarm tags.

Preparation

First an unused level input tag has to be created and then assigned to 'device' VirtAlarm. Go through the menus as below:

1. First a new tag must be created. From the root menu make selections `Editors` then `Tags` then `Edit` then `New`. A list of unused tags will now be displayed. Pick one, and reply yes to the prompt to create this tag. Choose type level input (L.I.).
2. You then supply a new tag name, and tag description. For example, tag number 90 is created, with name `VALARM1`, and description `Too much smoke`. Press escape <ESC> to finish creating the new tag.
3. Now that we have a new L.I. tag entry, we have to specify it to be a virtual alarm. From the root menu make selections `Editors` then `Tags` then `Edit` then `Change` then `90:VALARM1` then `Device`. You will have a choice of some actual devices, and either virtual math or virtual alarm devices. Select `VirtAlarm` and press `<return>`
4. As for the device `Channel`, you do not have to enter any, but type in `0` anyhow.
5. Finally, like all other level input tags, we must select an acquisition mode — from menu `Mode` select `Replace` mode.

Entering the Conditional Expression

Supposing we want an alarm to be triggered if:

(ATORQ < 42.5) AND (ENGSP > 300.0) AND (FUELT <= 23)

where ATORQ, ENGSP and FUELT are the values of level input tags. Then programming the expression involves the entry of 9 items: 3 each of tag number, comparison type, and value. This will become clear shortly, but first note that entry of the conditional expression for a virtual alarm tag takes place immediately after selecting VirtAlarm as a hardware device. If you wish to examine or alter it, pick menu items: **`Editors`** then **`Tags`** then **`Edit`** then **`Change`** then **`NN:PICKTAG`** then **`Device`** then **`VirtAlarm`**.

There are three level input tags, say *tag*1, *tag*2 and *tag*3, and three conditional operators, say *op*1, *op*2, and *op*3, as well as three constants, say *c*1, *c*2, and *c*3, to make up the virtual alarm expression. The tags *tag*1, *tag*2, and *tag*3 can only be level input (LI) tags. If you choose any other type of tag, then that one of three conditions (*tag op c*) will be ignored, and does not influence the alarm. The operators *op*1, *op*2, and *op*3 can be anyone of the following comparison operators:

<=, <, =, >, >=.

Where <= is the less than or equal to operator and >= is the greater than or equal to operator. The above tags, operators and constants form the following virtual alarm expression:

(tag1 op1 c1) AND (tag2 op2 c2) AND (tag3 op3 c3)

If a tag VALARM1 was programmed with the expression

(ATORQ < 42.5) AND (ENGSP > 300.0) AND (FUELT <= 23)

then tag will have a value of 1.0 if the above expression is true and 0 if false, provided that VALARM1's gain is set to 1.0 and offset is set to 0. If VALARM1's gain is set to 20 and offset is set to 5 then if the above expression is true the tag will have a value of 25 otherwise it will have a value of 5. As will be seen in the next section, to cause an alarm event, you simply set VALARM1's Lo, LoLo, Hi, HiHi alarm limits appropriately.

To enter the above expression, we use the menu items **`T1`**, **`OP1`**, **`C1`**, etc., by choosing **`Editors`** then **`Tags`** then **`Edit`** then **`Change`** then **`NN:MYTAG`** then **`Device`** then **`VirtAlarm`** then **`T1`**, etc. See table below for entries:

tag1	op1	c1	tag2	op2	c2	tag3	op3	c3
ATORQ	<	42.5	ENGSP	>	300.0	FUELT	<=	23

If only a two tag multi-level alarm is desired just enter a S.I. or S.O. or L.O. tag for one of the three tags. This eliminates that conditional from the three conditionals making up the multi-alarm alarm.

Triggering an Alarm

Multi-level alarms are programmed just like any other level input tag. The virtual alarm tag can be treated as a level input tag that has *only two values*. If the programmed alarm expression is true then it has the value *gain* + *offset* and if it is false it has the value of just the *offset*. If the gain is set to 1 and the offset is set to 0 then the VirtAlarm tag can have value of 1 or 0. Then set the alarm limits HiHi, Hi, Lo, and LoLo all to zero. Make sure alarming is enabled for both the system and the virtual tag (see regular alarm setting). This will trigger a HiHi alarm when the virtual alarm expression is true.

If an alarm action command is programmed for the HiHi alarm setting , then when the virtual alarm expression is true and alarming is on, the HiHi action will be executed.

Index

`.CFG` file 27
`.ENV` file 27
`.HLP` file 27
`.MAC` file 50
`.SCR` file 44
`.TAG` file 34
`.TMQ` file 57
`CONTROL` 5
`DATABUFF` 5
`L3MACROS` 5
`PCC32XXX` 5
AUTOMATIC mode **29**
MANUAL mode **29**
MONITOR mode 14, **29**
OFF mode **29**
action commands 11, 12, **47**, **78**
alarm event 53
alarm log 14, 26
alarm settings 53
alarm system 19, **53**
alarm trigger points 53
chaining, time queue 60
colour, screen output 45
environment 27
error log 14, 26, **47**
error messages **74**
error, in PID 18
esc key 9
filter, digital low pass **84**
format, screen output 45
function keys **13**
gain 36, 37
graphics 14-16, **62**
help information 27
historical trending 16, 63
indicators 8
Info menu **31**
information, system 31, 54
level output, changing 16, 39
load, in PID control 17
logging 65-67
logging interval 13
macro buffer 21, 26, 50
macro editor **50**
macro file 50
macros 20
maximum limit 37
menu system 8-12, 27
modes, PC Control operation**29**
nesting, macros 52, 76
offset 36, 37
paste, screen editor 45
PID control **17**
PID control algorithm **82**
ramp period 19, 43
ramping, set point 19, 43
rate, PID calculation 37
reporting 63-65
root menu 9
sample period, trend 61
screen editor 16, **44**
screen file 5, **44**
set point 17, 39
state output, changing 16, 39
syntax, checking **47**, 55
tag 8, 26
tag file 5, 26, **34**
tag structure 26, 34
tag values 61
tags editor **34**
target set point 19, 43
time queue 21
time queue buffer 26, 57
time queue editor **57**
trend buffer 26, **61**
trending interval **62**
Tune menu **39**
tuning, PID control loop 40-43
virtual device tags **68**
XY/Bar graphing 14, 63